VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

DIPLOMOVÁ PRÁCE

Brno, 2017 Bc. Marek Kudláček

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ
DEPARTMENT OF TELECOMMUNICATIONS

SYSTÉM PRO SPRÁVU A MONITORING PC
MANAGEMENT AND MONITORING SYSTEM FOR COMPUTER LAB

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. Marek Kudláček

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. Pavel Mašek

BRNO 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní obor **Telekomunikační a informační technika**
Ústav telekomunikací

Student: Bc. Marek Kudláček ***ID:*** 146881
Ročník: 2 ***Akademický rok:*** 2016/17

NÁZEV TÉMATU:

Systém pro správu a monitoring PC

POKYNY PRO VYPRACOVÁNÍ:

V rámci teoretické části diplomové práce bude proveden rozbor nástrojů pro správu a monitoring PC. V navazující praktické části bude přistoupeno k implementaci vybraného nástroje s cílem vytvoření uceleného systému pro vzdálenou správu PC, monitoring stavu těchto PC a následně celé učebny. Řešení bude vytvořeno jako nástavbový modulární systém s webovým rozhraním, který umožní správu z lokální nebo veřejné sítě.

DOPORUČENÁ LITERATURA:

[1] HART, Daniel. 2015. Ansible Configuration Management - Second Edition. Packt Publishing - ebooks Account. ISBN 978-1785282300.

[2] FRANK, Felix a Martin ALFKE. 2015. Puppet 4 Essentials - Second Edition. 2nd. Packt Publishing - ebooks Account. ISBN 978-1785881107.

Termín zadání: 1.2.2017 ***Termín odevzdání:*** 24.5.2017

Vedoucí práce: Ing. Pavel Mašek
Konzultant:

doc. Ing. Jiří Mišurec, CSc.
předseda oborové rady

ABSTRAKT

Diplomová práce se zabývá systémem pro hromadnou správu a monitoring PC. Je zde uveden stručný popis nástrojů Puppet, Chef a Ansible, které tuto funkcionalitu umožňují. Celková problematika systému je řešena pomocí nástroje Ansible. Součástí je vytvořený scénář pro vzdálenou správu učebny, podle předem stanoveného zadání. Systém zajišťuje vzdálenou instalaci a konfiguraci aplikací, vytváření uživatelů, editaci souborů, aktualizace systému a mnohé další. Pro tento systém je vytvořena webová aplikace, díky která může být systém ovládán pomocí webového prohlížeče, jak z lokální, tak i veřejné sítě. Součástí práce jsou praktické ukázky řešené problematiky.

KLÍČOVÁ SLOVA

Ansible, Apache, DNS, HTTP, HTTPS, Java, konfigurace, Linux, moduly, playbooky, síť, Spring, SSH, Windows, WinRM, vzdálená správa

ABSTRACT

This master's thesis deals with a system for PC mass management and monitoring. There is a brief description of Puppet, Chef and Ansible, which are tools enabling this functionality. The overall system issues are solved by Ansible. This thesis also includes a script created for a classroom's remote management according to a beforehand fixed assignment. The system enables remote installation and configuration of applications, creation of users, editing the files, update of the system and many more. A web application was created for this system. Through this application, the system can be controlled via web browser, both from local and public network. Practical examples of the issue dealt with is also part of the thesis.

KEYWORDS

Ansible, Apache, configuration, DNS, HTTP, HTTPS, Java, Linux, modules, network, playbooks, Remote management, Spring, SSH, Windows, WinRM

KUDLÁČEK, Marek *Systém pro správu a monitoring PC*: diplomová práce. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2017. 150 s. Vedoucí práce byl Ing. Pavel Mašek

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma „Systém pro správu a monitoring PC“ jsem vypracoval(a) samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor(ka) uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil(a) autorská práva třetích osob, zejména jsem nezasáhl(a) nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom(a) následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autora(-ky)

PODĚKOVÁNÍ

Chtěl bych moc poděkovat vedoucímu diplomové práce panu Ing. Pavlu Maškovi za skvělé odborné vedení, nesčetné konzultace plné podnětných návrhů a za rychlé reagování při řešení problémů spjatých s diplomovou prací. Také bych chtěl moc poděkovat celé své rodině za psychickou a finanční podporu, která mi umožnila studium na vysoké škole, kterou je právě Vysoké učení technické v Brně. Velké díky patří i mé přítelkyni Bc. Pavle Tetivové, která se mnou po čas celého studia sdílela úspěchy i neúspěchy a vždy dokázala udržet mojí hlavu klidnou.

Brno ....

podpis autora(-ky)

Faculty of Electrical Engineering
and Communication
Brno University of Technology
Purkynova 118, CZ-61200 Brno
Czech Republic

http://www.six.feec.vutbr.cz

PODĚKOVÁNÍ

Výzkum popsaný v této diplomové práci byl realizován v laboratořích podpořených z projektu SIX; registrační číslo CZ.1.05/2.1.00/03.0072, operační program Výzkum a vývoj pro inovace.

Brno . . . . . .

. . . . . .

podpis autora(-ky)

EVROPSKÁ UNIE
EVROPSKÝ FOND PRO REGIONÁLNÍ ROZVOJ
INVESTICE DO VAŠÍ BUDOUCNOSTI

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

ÚVOD

Obsahem diplomové práce je systém pro hromadnou správu a monitoring PC. Systémem je umožněn vzdálený dohled a správa zařízení uvnitř lokální sítě. Systém zprostředkovává komunikaci se vzdáleným zařízením nebo celou skupinou zařízení. Z jednoho místa tak lze obstarávat velké množství úkonů, kterými mohou být například instalace aplikací, spouštění skriptů, vytváření uživatelů, aktualizace operačních systémů, aktualizace aplikací, zjišťování informací o vzdáleném zařízení a spousty dalších.

Hlavní výhodou tohoto systému je všestrannost. Dokáže komunikovat jak s operačními systémy Windows (Windows 7 až 10, Windows Server 2008 až 2016), tak i s operačními systémy na bázi Linuxu (CentOS, Ubuntu, Debian a další).

Úvodní část 2 pojednává o dostupných možnostech jak výše zmíněný systém realizovat. Jsou diskutovány nástroje jako Puppet, Ansible nebo Chef. U těchto nástrojů je vytvořen stručný popis, seznámení s funkčností a rozdíly mezi technologiemi.

Dále 3.1 je vytvořen scénář učebny dle stanoveného zadání. Scénář ukazuje reálnou situaci každodenních úkonů správce sítě (instalace aplikací, tvorba uživatele, editace souborů, atd.). V diplomové práci je problematika scénáře řešena pomocí nástroje Ansible. Úkolem Ansible je zajistit hromadnou správu počítačů v již zmíněné učebně. Ke každému z řešených problémů je sepsáno teoretické pojednání s následnou ukázkou praktické implementace A.3.

V poslední části diplomové práce 4 je pro nástroj Ansible vytvořena webová aplikace, díky které je možné nástroj ovládat pomocí webového prohlížeče jak z lokální sítě, tak i z veřejné sítě. V textu jsou obsaženy nástroje potřebné pro vývoj aplikace 4.2.1 jako například *Maven*, framework *Spring* a vývojové prostředí *Spring Tool Suit*. Každý nástroj je náležitě představen a je objasněn jeho význam při vývoji webové aplikace. Potřebné úkony pro zprovoznění nástrojů jsou popsány v příloze A.4.1.

Webová aplikace musí být veřejně dostupná, a proto jsou v práci představeny i nástroje potřebné pro samotný provoz webové aplikace 4.2.2. Čtenář se tak dozví jak správně nakonfigurovat veřejný DNS server, síťový a osobní firewall nebo webový server *Apache*.

Jelikož je v dnešní době velmi častou otázkou bezpečnost, tak je i v diplomové práci brán na tuto problematiku ohled. Je představena ukázková konfigurace webového serveru Apache A.4.2 s použitím standardního nezabezpečeného protokolu HTTP, kdy je předvedeno jednoduché odposlechnutí přihlašovacího jména a hesla při přístupu do webové aplikace. Pro hlavní provoz aplikace je však využit zabezpečený protokol HTTPS, který opět obsahuje praktickou ukázku konfigurace a pokus o odposlechnutí hesla. Aby byl přístup na webovou aplikaci pro uživatele opravdu

důvěryhodný, tak je použit certifikát ověřený certifikační autoritou. V diplomové práci je tak obsažen popis a kroky potřebné k získání certifikátu.

Koncovým bodem diplomové práce 4.3 je samotná webová aplikace *Ansible-web-app*. Je zde objasněna struktura aplikace s vysvětlením jednotlivých funkcí. Obsažené funkce aplikace plně nahrazují terminálovou správu nástroje Ansible.

1 SÍŤ A JEJÍ SPRÁVA

V dnešní době se již komunikační sítě neskládají z několika počítačů, popřípadě serverů, ale tvoří ucelené skupiny s desítkami až stovkami stanic komunikujícími s různými typy serverů od poštovních až po databázové nebo aplikační. Takové množství zařízení vyžaduje až několik systémových správců, kteří využívají ke své práci řadu nástrojů pro dohled a správu nad takovou to sítí, nemluvě o množství odlišných konfiguračních syntaxí pro různé operační systémy. Správci sítě jsou tak nuceni ovládat tyto nástroje a umět je používat hned pro několik různých operačních systémů.

Jako příklad může posloužit instalace webového prohlížeče Mozilla Firefox pomocí nástroje powershell[1] pro stanice s operačním systémem Windows, a pomocí terminálu[2] pro stroje s operačním systémem Linux, například distribuce Ubuntu a CentOS. Následující příkazy jsou psány pod uživatelem admin pro Windows a root pro Linux.

```
PS C:\WINDOWS\system32> Install-Package -name firefox
```

```
[root@ubuntu~]# apt-get install firefox
```

```
[root@centos~]# yum install firefox
```

V tomto jednoduchém příkladu lze vidět, jak rozdílné jsou syntaxe pro pouhé nainstalování webového prohlížeče Mozilla Firefox. Takovýchto odlišností na jednotlivých operačních systémech lze nalézt více. Je jisté, že si správci sítě vystačí s těmito znalostmi, pokud budou mít na starosti síť s desítkami počítačů a několika servery. Situace bude mít však dopad na větší chybovost, jelikož do zařízení bude zasahovat několik správců, kteří nebudou mít jednotný styl při nastavování.

Dalším limitujícím faktorem je čas. Pokud by bylo nutné v síti konfigurovat každé zařízení jednotlivě, tak by to bylo velmi časově náročné. Nejvýhodnějším řešením je tedy využívat specifických orchestračních nebo automatizačních nástrojů, díky kterým by bylo možné aplikovat potřebná nastavení na definované skupiny stanic ve stejný čas. Takovýto nástroj poskytuje okamžitou notifikaci, zda použité nastavení bylo na daném zařízení úspěšně aplikováno, či ne. Dále lze zobrazit potřebné informace o jednotlivém zařízení či celé skupině (například IP adresu, název zařízení,

[1] powershell – nástroj pro ovládání systému Windows pomocí textových příkazů. Nástupce příkazové řádky Command Line.

[2] terminál – obdoba powershellu, pro unix a linux systémy.

stav disku, instalované aplikace a další informace), aniž by bylo potřeba se na zařízení nejprve připojit a poté zadat sadu příkazů, kterými by byly tyto informace objasněny.

S využitím výše popsaného nástroje by bylo možné eliminovat chyby při manipulaci se sítí, byla by urychlena správa a monitorování, také by byl ušetřen čas pro zdokonalování a rozvoj již stávající infrastruktury sítě.

2 MOŽNOSTI SPRÁVY SÍTĚ

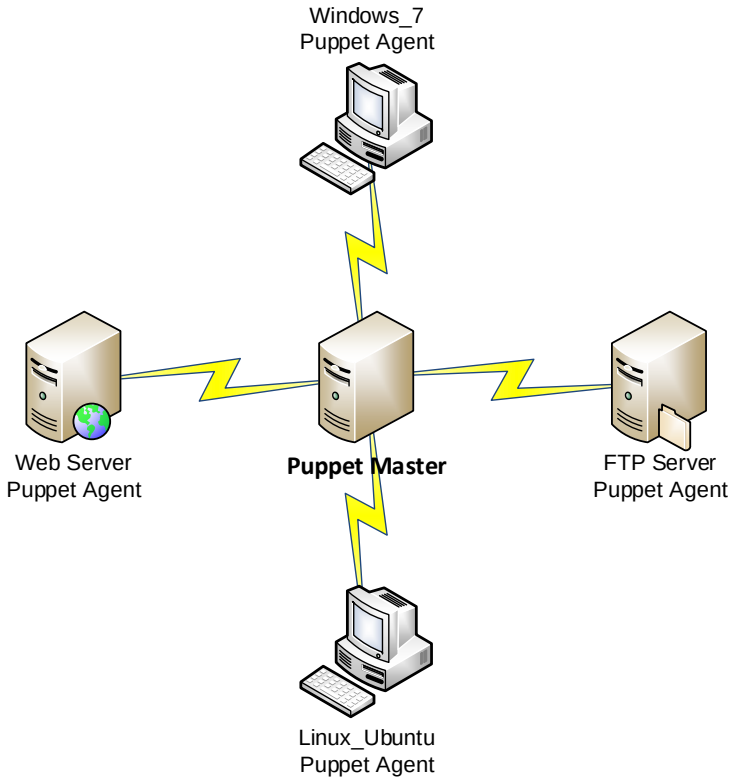
V Kapitole 1 již bylo řečeno, že je dosti obtížné dohlížet a spravovat celou síť s velkým množství zařízení pracujících s odlišnými systémy. Na trhu existuje několik nástrojů, které dokáží automatizovat správu sítě. Jsou to převážně serverové aplikace, které však mohou pracovat i na virtuálních strojích. Ve zkratce budou představeny některé z těch nejznámějších CM (Configuration Management) nástrojů na trhu.

- **CFEngine**
  Jeden z nejstarších automatizačních nástrojů. Nejnovější verze je CFEngine3. Využívá jazyk C, komunikaci typu klient–server. Vytváří sliby, což jsou jednotlivé příkazy. Skupiny slibů mohou utvořit tzv. balík, který je následně používán na požadované stroje. Využívají jej například společnosti Samsung a Panasonic [1].
- **Puppet**
  Open source nástroj, využívající jazyk Ruby, také typu klient–server. Ke své práci využívá zdroje, ve kterých jsou definovány stavy. Každý stav je souhrn příkazů. Celkový stav specifikující vzdálený systém je uložen v manifestu, kde manifest je složen z potřebného počtu zdrojů. Využívají jej například společnosti Sony a NASA [2].
- **Chef**
  Tak jako Puppet je Chef open source nástroj založený na Ruby, typu klient–server. Nástroj umožňuje tvorbu receptů, ve kterých jsou obsaženy příkazy. Z několika receptů se pak vytvoří kuchařka, která je aplikována na definované stanice. Tato kuchařka specifikuje konečný stav vzdálené stanice. Využívají jej například společnosti Hewlett Packard Enterprise a FastRobot [3].
- **Ansible**
  Open source nástroj napsán ve skriptovacím jazyku Python, nevyužívá komunikaci typu klient–server, ale typu push, šifrovanou pomocí SSH komunikace na vybrané stroje. Pro svou správu využívá jazyk YAML (Ain't Markup Language). Stavy stanic jsou obsaženy v hrací knize, která obsahuje určité role. Každá role specifikuje sled příkazu, popisující požadovaný stav stroje. Využívají jej například společnosti Cisco Systems a Hewlett Packard [4].

V následujícím textu bude blíže objasněna funkce těchto, v poslední době nejrozšířenějších, nástrojů. Budou jimi Puppet 2.1, Ansible 2.2 a Chef 2.3.

2.1 Puppet

Puppet pro svou práci využívá komunikaci klient–server. Klient, neboli *puppet agent*, je stroj obsažen v infrastruktuře, který je spravován a jsou na něm prováděny potřebné konfigurace. *Puppet master* reprezentuje hlavní server, kterým je dohlíženo na celou síť. Předává agentům potřebné informace o jejich nastavení, popřípadě informace o jejich stavu. Schéma lze vidět na Obr. 2.1. Samotný Puppet master může být i agentem. Každý z agentů disponuje určitými zdroji, kterými jsou například různé instalované aplikace (Google Chrome, Putty, MS Office a další), soubory, oprávnění k souborům nebo adresářům, atd.



Obr. 2.1: Schéma sítě Puppet.

Pro tvorbu specifické konfigurace Puppet využívá *moduly*, *manifesty* a *třídy*, o nichž bude více pojednáno v sekci 2.1.1. Na Puppet masteru je uložena potřebná konfigurace v takzvaném manifestu což, zjednodušeně řečeno, je souhrn příkazů. V Puppet manifestu jsou pak specifikovány tyto zdroje a jejich stavy. Stavem se rozumí, co se má s daným zdrojem vykonat, zda má být například daný balíček nainstalován nebo zda má být nastaveno určité umístění souboru. Pokud se stav

na agentu shoduje se stavem, který je obsažen v modulu, tak se žádná změna neprovádí a pokračuje se na další příkaz.

Správa sítě může být provedena těmito způsoby:

- **Jednorázově:** na agenty bude v síti aplikována potřebná konfigurace dle potřeby.
- **Automaticky:** každý z agentů se v určitém časovém intervalu (přibližně 30 minut) připojuje na puppet mastera a dotazuje se jej na svou konfiguraci.

Puppet umožňuje spravovat jak Linuxové, Unixové, tak i Windows stroje. Windows stroje mohou být pouze agenty, nikoli však Puppet masterem. Puppet master tedy může běžet pouze na Linux, či Unix systémech. Puppet nevyužívá grafického rozhraní, a kromě dashboardu, kde se zobrazují stavy a zprávy o změnách konfigurací, je plně konzolový.

2.1.1 Puppet konfigurace

Potřebné požadavky na instalaci, spuštění a konfiguraci Puppetu jsou přehledně zdokumentovány na docs [5] stánkách Puppetu. Proto budou ukázány pouze základní kroky potřebné pro zdárnou funkčnost systému.

Instalace Puppet master

Puppet server může být nainstalován pouze na *NIX strojích. Výběr je tedy možný například z distribucí Debian, Ubuntu, SuSE, CentOS nebo Red Hat Enterprise Linux. Dále je nutné, aby byly zohledněny systémové požadavky, které musí daný stroj splňovat. Minimálním požadavkem na verzi Puppet 4.7 je dvoujádrový procesor s architekturou x86_64 a 1 GB RAM. Ve výchozím nastavení jsou Puppet masterem vyhrazeny 2 GB operační paměti.

Nejprve je naistalován balíček *puppetserver*.

```
[root@puppet-server ~]# yum install puppetserver
```

Po úspěšné instalaci je spuštěna služba.

```
[root@puppet-server ~]# systemctl start puppetserver
```

Instalace Puppet agent

Před instalací agenta je nutné zvážit, zda bude agent pracovat na *NIX nebo Windows stroji. V našem případě bude provedena ukázka instalace na stroji s operačním systémem Windows 7. Instalace může být dvojího typu:

- **Grafický průvodce**.
- **Příkazová řádka**, jak je znázorněno níže. Tímto příkazem je umožněna automatická instalace. Soubor msi má specifické vlastnosti, které je potřeba v příkazu specifikovat. Celkový souhrn vlastností je dostupný na stránkách Puppetu [6].

```
C:\WINDOWS\system32> msiexec /qn /norestart /i puppet-agent-1.7.1-x64.msi
```

Před stažením msi (instalační balíček pro systémy Windows) souboru ze stránek [7] je zapotřebí zohlednit typ architektury, kterou daný operační systém využívá. Po instalaci je potřeba Puppet agenta ještě řádně nastavit, především specifikovat FQDN (Fully Qualified Domain Name) Puppet mastera, v opačném případě by komunikace nemusela být funkční.

2.1.2 Prostředí

Prostředí, neboli *environment*, umožňuje rozdělit agenty do skupin. V každém prostředí mohou být použity odlišné manifesty a moduly. Pokud by nebylo zapotřebí rozdělení agentů do skupin, tak by bylo ve výchozím stavu použito prostředí s názvem *production*. V tomto prostředí jsou specifikováni všichni agenti, kteří se spojí s Puppet master serverem.

2.1.3 Manifest

Manifest představuje sadu příkazů, které se mají na určitém agentovi nebo skupině agentů provést. Manifest může být vytvořen jak na Puppet masteru, tak i na Puppet agentu. Soubor manifest musí mít příponu *.pp*, jak bude ukázáno na příkladu manifestu *copy_delete.pp*.

Správně vytvořený manifest může být aplikován přímo na systém, kdy se zadané příkazy okamžitě vyhodnotí, nebo v testovacím režimu pomocí parametru *noop*, kterým je ověřena správná funkce manifestu. Tímto parametrem jsou eliminovány možné chyby. Při prvotním použití manifestu je doporučeno použití parametru noop, aby bylo možné předejít chybám, kterými by mohlo dojít k dočasnému nebo i úplnému vyřazení z provozu. Manifest může být psán v libovolném textovém editoru. Důraz musí být však kladen na uložení souboru jako *Ruby file*, jelikož je Puppet

napsán v programovacím jazyce Ruby. Ukázkový příklad manifestu je zobrazen níže.

Manifest copy_delete.pp

```
$source_dir = 'c:\\puppet\\SOURCE':
$destination_dir = 'c:\\puppet\\DESTINATION':
file { $destination_dir :
        ensure  => 'directory',
        source  => "file://${source_dir}",
        recurse => true,
        before  => File[$source_dir],
}
file { $source_dir :
        ensure  => 'absent',
        purge   => true,
        recurse => true,
        force   => true,
}
```

Po spuštění tohoto manifestu příkazem níže dojde k překopírování obsahu adresáře *SOURCE* do adresáře *DESTINATION*, poté se adresář SOURCE smaže. Manifest musí být spuštěn z adresáře, ve kterém je soubor obsažen.

```
PS C:\puppet> puppet apply -t .\copy_delete.pp
```

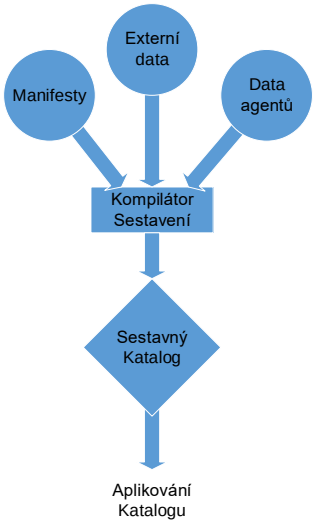
2.1.4 Katalog

Hlavním účelem Puppetu je udržovat systémové zdroje zařízení ve stavu, které definují právě manifesty, tedy soubory s příponou *.pp*. Manifesty předurčují cílový stav zařízení. Stav systému se pak neskládá z jednoho, ale hned několika manifestů. Aby manifesty mohly být použity, je zapotřebí *katalog*. Je-li použit Puppet v režimu apply, tak je vytvořen katalog, který se skládá z kompilací 3 zdrojů:

- **Zdroj dat poskytnutých z agenta:** zde je obsaženo jméno agenta, certifikát, informace o agentu.
- **Externí data:** mohou být, pokud to je při kompilaci vyžadováno, použita i data z dalších zdrojů jako například Puppet databáze.
- **Puppet manifesty, šablony, moduly a další:** tento zdroj informací je nejdůležitější složkou při vytváření katalogu, jelikož jsou v něm obsaženy informace o systémovém stavu zařízení. Je složen z vytvořených manifestů, modulů

stažených z Puppet Forge [8] (repozitář modulů psaný komunitou Puppet) nebo svých vlastních.

Grafické schéma vytvořeného katalogu je znázorněno na Obr. 2.2.



Obr. 2.2: Grafické znázornění vytváření katalogu.

2.1.5 Zdroje

Aby bylo možné deklarovat systémové zdroje na cílovém zařízení, musí být tyto zdroje specifikovány v patřičném manifestu. V sekci 2.1.3 bylo možno vidět ukázku manifestu s názvem *copy_delete.pp*, který využívá zdroje k definování stavů systému. Zdroje jsou děleny například na zdroje pro:

- **Jádro systému:** exec, group, host, package.
- **Windows:** acl, powershell, reboot, registry.

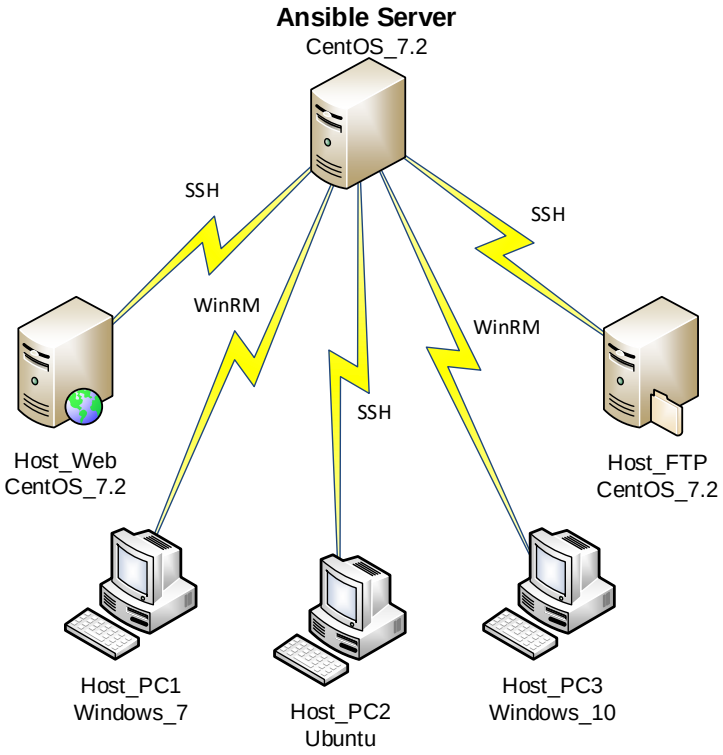
Příklad zdroje pro zmíněný manifest je uveden níže.

```
file { $destination_dir :
        ensure  => 'directory',
        source  => "file://{source_dir}",
        recurse => true,
        before  => File[$source_dir],
}
```

Je zde použit zdroj *file*, který je určený pro jádro systému. Zdroj *file* využívá atributy (stavy): *ensure*, *source*, *recurse* a *before*. Každý atribut pak má svou vlastnost, jako je například vlastnost *directory* u atributu *ensure*. Veškeré další informace o zdrojích jsou obsaženy na stránkách [9].

2.2 Ansible

Ansible je orchestrační nástroj, který ke své práci nepotřebuje žádné předinstalované agenty na hostovaných stanicích. Open source verze je plně konzolová a grafické rozhraní zde není využíváno. Grafické prostředí je součástí pouze placené verze Ansible Tower. Ke své komunikaci využívá protokol SSH (Secure Shell protokol pro šifrovanou komunikaci), který slouží ke komunikaci mezi linuxovými a unixovými stroji, nebo služby WinRM (umožňuje komunikaci odlišných typů zařízení s odlišnými typy operačních systémů). Kořenem celé infrastruktury je linuxový server, na kterém je Ansible nainstalován. Z tohoto serveru jsou pomocí SSH nebo WinRM spravováni vzdálení hosté, jak lze vidět na Obr. 2.3.



Obr. 2.3: Schéma sítě Ansible.

Práce s nástrojem Ansible je mnohem snadnější a rychlejší, jelikož odpadá starost s časově náročným instalováním a konfigurováním potřebných klientů. Stačí pouze vygenerovat dvojici SSH klíčů na serveru Ansible a ty rozkopírovat na hostující stanice, v případě Linux nebo Unix strojů. Pokud bude vyžadována správa Windows strojů, musí být nastavena na serveru komunikaci pomocí WinRM a na příslušných hostujících strojích s operačním systémem Windows je vygenerován *self-signed* [1] certifikát.

Na serveru je následně vytvořen seznam hostů, na které bude aplikován sled příkazů, které mají být provedeny. Každý host v seznamu je specifikován, buď IP adresou nebo FQDN (Fully Qualified Domain Name) názvem.
Příkazy pak mohou být vytvořeny pomocí modulů viz sekce 2.2.4 nebo skriptů, které jsou specifické pro daný operační systém.

2.2.1 Konfigurace Ansible

Potřebné požadavky na instalaci, spuštění a konfiguraci Ansible, jsou přehledně zdokumentovány na docs stránkách Ansible [10]. Budou tedy jen ve zkratce ukázány základní kroky potřebné pro zprovoznění systému.

2.2.2 Instalace Ansible serveru

Ansible server může být nainstalován pouze na *NIX strojích. Systémové požadavky pro instalaci Ansible server jsou bohužel specifikovány pouze pro Ansible Tower, který je, jak již bylo zmíněno dříve, placenou grafickou verzí Ansible. Na stránkách Ansible v části požadavky [11] je možné zjistit potřebnou velikost minimálně 2 GB nebo doporučené 4 GB RAM, dále pak 20 GB místa na disku a operační systém musí podporovat 64-bit instrukce. Open source Ansible si tedy vystačí s těmito požadavky. Ansible server může být nainstalován na jakoukoliv *NIX distribuci, která obsahuje Python 2.6 (programovací/skriptovací jazyk) a vyšší.

Instalace pro účely diplomové práce bude demonstrována na CentOS 7.2.1511. Nejprve bude nainstalován EPEL repositář, který je již součástí CentOS 7.2. Z tohoto repozitáře bude Ansible stažen a následně nainstalován.

```
[root@ansible-server ~]# yum install epel-release
```

Poté bude Ansible nainstalován.

```
[root@ansible-server ~]# yum install ansible
```

[1] self-signed - digitální certifikát, který podepsal sám jeho tvůrce.

2.2.3 Specifikace hostů a skupin

Jelikož Ansible nevyužívá ke své činnosti žádné agenty předinstalované na hostujících strojích, nebude potřebná žádná další instalace. Pro hostování *Unix strojů postačí funkční SSH komunikace. Pro hostování Windows strojů je zapotřebí minimálně verze powershell 3.0 a funkční WinRM komunikace.
Stroje, které budou spravovány, musí být specifikovány v souboru,

```
[root@ansible-server ~]# /etc/ansible/hosts
```

ve kterém seznam hostovaných stanic může vypadat následovně.

```
[windows]
windows7-pc ansible_ssh_host=192.168.173.131
windows10-pc ansible_ssh_host=192.168.173.133
NB10.mojedomena.local
PC20.mojedomena.local

[linux-servers]
DATABASERVER.mojedomena.local
WEBSERVER.mojedomena.local
```

Z výpisu je patrný seznam se čtyřmi stroji s operačním systémem Windows a dvěma linuxovými servery. Tyto stroje jsou rozděleny do skupin *[windows]* a *[linux-servers]*. Pro stroje, které nejsou obsaženy v interním DNS (Domain Name System), je použit příkaz *ansible_ssh_port*. Tento příkaz umožňuje spárování informací o doménovém názvu stroje a použité IP adrese, přímo v souboru */Ansible/hosts*. Určení hosta, na kterém budou definované příkazy vykonány, je specifikováno právě doménovým názvem.

2.2.4 Moduly

Moduly umožňují kontrolovat systémové zdroje stroje, jako například služby, balíčky, soubory, oprávnění, atd. Moduly v Ansible jsou jako zdroje v Puppetu. Modul může být spuštěn jako jednotlivý příkaz

```
[root@ansible-server ansible]# ansible -i hosts windows -m setup
```

nebo jako playbook, který může obsahovat i více modulů. Příkaz vypíše systémové informace z hostů obsažených ve skupině Windows pomocí modulu *setup*, který je zavolán parametrem *-m*. Každý modul má své specifické vlastnosti a nastavení pomocí dalších parametrů, jak bude ukázáno v sekci Playbook 2.2.5. Seznam

všech dostupných modulů je obsažen na stránkách Ansible, v kategorii modules [12]. Pro vypsání rozšiřujících nastavení u jednotlivých modulů poslouží následující příkaz.

```
[root@ansible-server ~]# ansible-doc setup
```

2.2.5 Playbook

Playbook je soubor, který obsahuje skupiny her, sled příkazů, které se mají na daném hostu vykonat. Tyto hry se skládají z modulů. Playbook je soubor využívající jazyka YAML (Ain't Markup Language). Playbook má tedy vždy příponu *.yml*. Níže je zobrazen playbook pro instalaci aplikací s názvem *install_via_chocolatey.yml*.

```
- name: Windows_chocolatey module
  hosts: windows
  tasks:
    - name: Install firefox
      win_chocolatey:
        name: firefox
        state: present

    - name: Install putty
      win_chocolatey:
        name: putty.portable
        state: present
```

Playbookem je nainstalována skupina hostů s operačním systémem Windows s aplikacemi Mozilla Firefox a Putty. V playbooku je definovaná skupina hostů *host*, na kterých jsou úkony *tasks* vykonávány. V *tasks* jsou použity moduly *win_chocolatey*, které umožňují bezobslužnou instalaci aplikací na systémy Windows. Moduly využívají parametry *name* pro specifikaci aplikace, která bude nainstalována a parametry *state* pro určení stavu aplikace na systému hosta. Pro použtí playbooku slouží následující příkaz.

```
ansible-playbook install_via_chocolatey.yml
```

Playbook může disponovat obrovským množstvím informací, které se mají na hostující stanici provést. Pokud má být playbook co nejpřehlednější, tak lze využít role, viz níže.

2.2.6 Role

Ansible role pomáhají rozčlenit jednotlivé, stále se opakující, úkony do oddělených celků. Usnadňují tak čtení a umožňují použití dalších potřebných souborů, kterými mohou být například vlastní skripty *.sh*, *.ps* nebo soubory *.msi*, *.conf*, a jiné.

Role se skládá z adresáře, který má v sobě další podadresáře. Tyto podadresáře v sobě obsahují soubor *main.yml*, který opět jako playbook specifikuje sled operací, které mají být provedeny.

Hlavní *main.yml* soubor se nachází v adresáři *tasks*. Role *install_via_chocolatey*, může vypadat následovně. Jedná se o zjednodušení předchozího příkladu.

```
[root@ansible-server install_via_chocolatey]# ll
total 4
drwxr-xr-x. 2 root root   21 Nov  5 01:34 defaults
drwxr-xr-x. 2 root root    6 Nov  5 01:34 files
drwxr-xr-x. 2 root root   21 Nov  5 01:34 handlers
drwxr-xr-x. 2 root root   21 Nov  5 01:34 meta
-rw-r--r--. 1 root root 1328 Nov  5 01:34 README.md
drwxr-xr-x. 2 root root   21 Nov  5 01:34 tasks
drwxr-xr-x. 2 root root    6 Nov  5 01:34 templates
drwxr-xr-x. 2 root root   37 Nov  5 01:34 tests
drwxr-xr-x. 2 root root   21 Nov  5 01:34 vars
```

Samotný soubor *main.yml* v adresáři *tasks* vypadá následovně. Pomlčkami je specifikován každý začátek *main.yml* souboru v adresáři *tasks*, pro danou roli. Níže obsah *main.yml* souboru v $\backslash etc \backslash ansible \backslash roles \backslash testovaciRole \backslash tasks \backslash main.yml$.

```
---
# Install Mozilla Firefox 49.0.2.
        win_chocolatey:
        name: firefox
# Install Putty Portable
        win_chocolatey:
        name: putty.portable
```

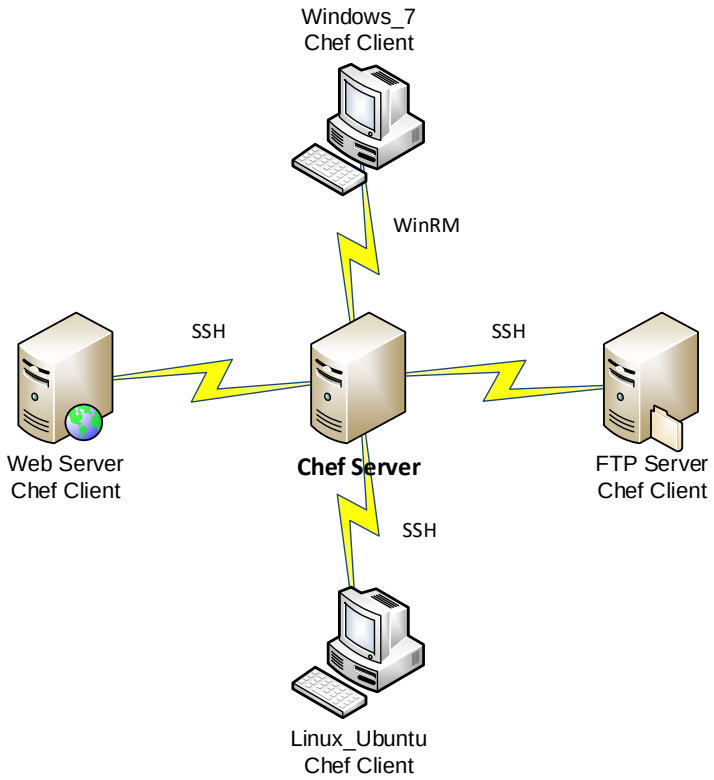
Obsah playbooku pro instalaci aplikací pak může vypadat takto.

```
- hosts: windows
  roles:
     - install_via_chocolatey
```

Role jsou velmi užitečnou pomůckou pro psaní opravdu rozsáhlých playbooků.

2.3 Chef

Chef je posledním automatizačním nástrojem, který bude v diplomové práci diskutován. Stejně jako Puppet je založen na komunikaci klient - server, obr. 2.4. Kořenovým prvkem je *Chef Server*, který dohlíží nad systémovými stavy *Chef Klientů*. Je tedy zapotřebí na každém zařízení nainstalovat klienta, který bude zajišťovat komunikaci s Chef serverem. Open source verze umožňuje obsluhovat Chef server i pomocí webového rozhraní, avšak správa sítě povoluje dohled maximálně nad 25 uzly, tedy klientskými stanicemi. Pokud by vznikla potřeba dohledu nad větší infrastrukturou, tak bude potřeba zakoupit licenci.



Obr. 2.4: Schéma sítě Chef.

Chef oproti Puppetu a Ansible dovoluje instalaci Chef serveru na zařízení s Windows Server 2012 R2. Jako předešlí konkurenti zajišťuje spolupráci se systémy *UNIX i Windows. Komunikace je zajištěna pomocí služby WinRM, využívající protokol SSH a self-signed certifikáty. Spolu s nástrojem *knife* (nástroj příkazové řádky, tvoří rozhraní mezi lokálním Chef repozitářem a Chef serverem).

2.3.1 Konfigurace Chef

Potřebné požadavky na instalaci, spuštění a konfiguraci Chef, jsou přehledně zdokumentovány na docs stánkách Chef [13]. Proto budou ukázány jen základní kroky potřebné pro zdárnou funkčnost systému.

2.3.2 Instalace Chef serveru

Již bylo zmíněno, že Chef server je podporován nejen na Red Hat Enterprise Linux, CentOS, Ubuntu, ale i na Windows Server 2012 R2. Je tedy prvním ze zmíněných automatizačních nástrojů, podporující i operační systém Windows. Chef má ze všech nástrojů největší požadavky kladené na systém.

Hlavním požadavkem na systém je podpora 64-bitových systému, 32-bitové verze nejsou podporovány. Další požadavky pro instalaci *standalone*:

- Čtyř jádrový procesor 2.0 GHz AMD 41xx/61xx nebo Intel Xeon 5000/E5,
- 4 GB v základu nebo 8 GB RAM pokud bude využívat služby Chef Anylytics,
- 20 GB volného prostoru.

Seznam dalších požadavků je dostupný na stránkách docs.chef v kategorii systémové požadavky [14].

Instalace bude opět demonstrována na CentOS 7.2.1511. Nejprve bude stažen instalační balíček z webu do adresáře *tmp*. Balíček pro danou platformu je obsažen na webu [15].

```
[root@chef-server tmp]# curl "https://packages.chef.io/stable/el/7/
chef-server-core-12.10.0-1.el7.x86_64.rpm"
```

Je nainstalován stažený balíček.

```
[root@chef-server ~]# rpm -Uvh /tmp/
chef-server-core-12.10.0-1.el7.x86_64.rpm"
```

Dále budou spuštěny služby potřebné pro Chef server.

```
[root@chef-server ~]# chef-server-ctl reconfigure
```

2.3.3 Instalace Chef klient

Instalace Chef klientů je rychlá, avšak následná konfigurace komunikaci mezi klientem a serverem je dosti komplikovaná. To platí jak pro klienta s operačním systémem Windows, tak i Linux. Budou objasněny základní kroky pro konfiguraci klienta Windows, zbylé body jsou podrobně popsány na webu [16].

V první řadě musí být správně nakonfigurován *Knife on Windows*, jehož konfigurace je dostupná zde [17].

Dále musí být stažen klient pro potřebnou edici Windows a následně může být spuštěna instalace. Nesmí být opomenuto, že musí být klient spuštěn jako služba.

```
PS C:\WINDOWS\system32> msiexec /qn /i C:\inst\
chef-client-12.4.3-1.windows.msi
ADDLOCAL="ChefClientFeature,ChefServiceFeature,ChefPSModuleFeature"
```

2.3.4 Zdroje

Zdroje jsou obdobou modulů u Ansible a zdrojů u Puppetu. Zdroji je popisován požadovaný stav systému. Využívají stejného principu jako předešlé automatizační nástroje. Na serveru jsou uloženy soubory s recepty viz sekce 2.3.5, které v sobě ukrývají sled příkazů poskládaných z jednotlivých zdrojů. Seznam všech dostupných zdrojů je k dispozici na webu docs.chef v kategorii *Resources* [18]. Níže je zobrazena podoba jednoduchého zdroje.

```
file '/tmp/motd' do
        content '!!!Vítejte na Chef serveru!!!'
end
```

Tímto zdrojem je popsán stav souboru *motd*, ve kterém má být uložen text: *!!!Vítejte na Chef serveru!!!*. Zdroj *file* má v sobě souhrn *akcí*, které může provádět. V tomto případě je využita akce *content*, kterou je zajištěn zápis obsahu textu do souboru motd.

2.3.5 Recepty

Recepty jsou soubory s příponou *.rb*, které v sobě obsahují seznam stavu zdrojů. V sekci 2.3.4 bylo ukázáno jak takový zdroj vypadá, tato syntaxe však musí být uložena do receptu. Zdroj bude tedy uložen do receptu *set_motd.rb*.

Recepty jsou obdobou manifestů z Puppetu a rolí z Ansible. Spuštěním následujícího příkazu, bude provedena změna na klientských stanicích.

```
[root@chef-server ~]# chef-client set_motd.rb
```

2.3.6 Kuchařka

Kuchařka může být použita pouze, pokud obsahuje požadované recepty. Kuchařkou je popisován koncový stav systému, který je specifikován v jednotlivých receptech.

Každým z klientů je zažádáno, v určitém čase, o kuchařku, podle které je pak daný systém nakonfigurován. Obdobným způsob je využíván u playbooků v Ansible a katalogů u Puppetu.

2.4 Srovnání Puppet, Ansible a Chef

V Tabulka 2.1 je vyobrazen souhrn nejdůležitějších vlastností Puppet 2.1, Ansible 2.2 a Chef 2.3 spolu s jejich rozdíly. Každý nástroj obsahuje svá pro a proti a je tedy důležité, aby tato fakta byla uvážena před samotným výběrem daného nástroje. Výčet vlastností poslouží k rychlému seznámení a umožní tak čtenáři zacílit svoji pozornost na ten nejpotřebnější z popisovaných orchestračních nástrojů. V této práci bude dále popsán způsob správy sítě pomocí Ansible, který byl vybrán na základě těchto kriterií:

- **Open source nástroj, umožňující správu neomezeného počtu stanic.**
- **Nejsou využíváni klienti.**
- **Nejsou kladeny zvýšené nároky na výkon serveru.**

Tab. 2.1: Srovnání Puppet, Ansible, Chef

Nástroj:	**Puppet**	**Ansible**	**Chef**
Vznik:	2005	2012	2009
Jazyk:	Ruby	Python	Ruby
Agenti:	Ano	Ne	Ano
Klienti Linux/Windows:	Ano	Ano	Ano
Server Linux/Windows:	pouze Linux	pouze Linux	Ano
Open source Web UI:	Ne	Ne	Ano (do 25 stanic)
Šifrovaná komunikace:	Ano	Ano	Ano
Potřebná znalost programování:	Ano	Ne	Ano
Dostupnost dokumentace:	Velmi dobrá	Velmi dobrá	Velmi dobrá

3 NÁVRH SCÉNÁŘE ŠKOLNÍ SÍTĚ

Další část diplomové práce bude zaměřena na ukázku praktického použití nástroje Ansible pro správu učebny ve školní síti VUT Brno, Fakulty elektrotechniky a komunikačních technologií. Nástroj musí být schopen vzdálené obsluhy všech stanic umístěných v učebně. Podrobnější informace jsou obsaženy v následující části textu.

3.1 Zadání

Učebna je složena ze stanic s operačním systémem:

- **Windows 10**,
- **Windows 7**,
- **Ubuntu 16.04**,
- **Debian 8**.

Na stanicích je zapotřebí vzdáleně spravovat instalace a odinstalace programů:

- **Microsoft office**,
- **Putty**,
- **Wireshark**,
- **webové prohlížeče Mozilla Firefox a Google Chrome**,
- **Microsoft Visual Studio**,
- **Total Commander**,

spolu s aktualizací na nejnovější verze. Musí být umožněno:

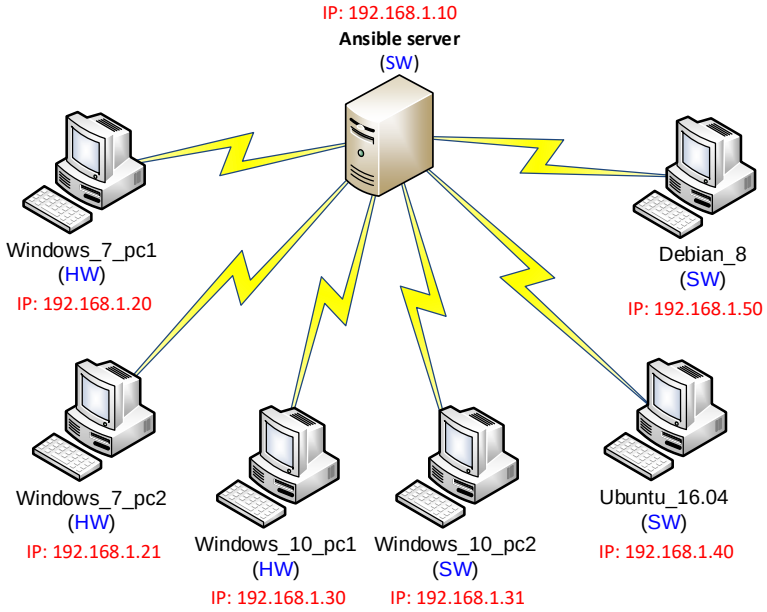
- **zobrazení informací o stanici**,
- **restart a vypnutí stanice**,
- **tvorba uživatelů**,
- **vytváření a mazání souborů**,
- **instalace aktualizací**.

Výše zmíněná nastavení musí být implementována s pomocí jediného nástroje.

3.2 Testovací síť

Je na uvážení každého administrátora, zda bude nástroj otestován přímo v ostrém provozu nebo zda raději budou provedeny zkoušky v testovací síti. V tomto případě bude zvolena druhá varianta, kdy bude vytvořena testovací síť. Základem pro nasazení CM nástrojů je fungující komunikační síť. Předpokladem následujícího textu bude seznámení čtenáře s možností hromadné správy sítě pomocí Ansible, nikoli konfigurace síťových prvků pro docílení fungující komunikační sítě.

Testovací síť bude využívat několik virtuálních stanic spolu s několika hardwarovými stanicemi. Logickým schématem sítě viz Obr. 3.1 je specifikována hierarchie komunikace. Stanice jsou rozděleny podle operačního systému a typu. Typem je myšleno, zda má stanice svůj vlastní hardware nebo zda je vytvořena pomocí virtualizace.



Obr. 3.1: Logické schéma testovací sítě.

Polovina stanic i samotný ansible server jsou vytvořeny pomocí virtualizačního nástroje VMWare vSphere 6.0, který je zprovozněn na fyzickém stroji HP ProLiant DL360 G6, s následujícími parametry:

- 4 jadrové CPU, Intel(R) Xeon(R) E5504, na frekvenci 2.00 GHz,
- 12 GB RAM,
- 500 GB pevný disk.

Pro operační systém Windows 7 byly vyčleněny dva hardwarové stroje Dell Vostro 260s. Pro systém Windows 10 jeden virtuální stroj a jeden hardwarový stroj Dell Optiplex 3020.

Parametry jednotlivých stanic jsou znázorněny v Tabulce 3.1. Doplňující informace o VMware vSphere 6.0 jsou dostupné v příloze A.1.

3.2.1 Nasazení Ansible do testovací sítě

Pro orchestrační nástroj Ansible je vyžadován systém s operačním systémem *Unix nebo Linux, jak již bylo zmíněno v Kapitole 2.2. Pro účely testovací sítě byl zvolen 64

Tab. 3.1: Parametry virtuálních a hardwarových strojů v testovací síti.

STROJE:	CPU	RAM	HD
VMware vSphere 6.0			
Ansible server	E5504 (1 jádro)	2 GB	50 GB
Win_10_pc2	E5504 (1 jádro)	4 GB	32 GB
Ubuntu_16.04	E5504 (1 jádro)	1 GB	32 GB
Debian_8	E5504 (1 jádro)	1 GB	32 GB
Dell Vostro 260s			
Win_7_pc1	E5400 (2 jádro)	8 GB	500 GB
Win_7_pc2	E5400 (2 jádro)	8 GB	500 GB
Dell Optiplex 3020			
Win_10_pc1	i5-4590 (2 jádro)	8 GB	256 GB

bitový operační systém CentOS 7.2.1511, který byl nainstalován na stroj s názvem *Ansible server*, jak bylo znázorněno na Obr. 3.1.

Instalace a konfigurace Ansible

Potřebné kroky pro základní nastavení a instalaci byly zmíněny v sekci 2.2.2.

Konfigurace Ansible serveru pro komunikaci s Windows

Po instalaci Ansible na server bude vytvořen adresář *group_vars* se souborem *windows.yml*, ve kterém je specifikována potřebná konfigurace pro navázání spojení Ansible serveru se stanicemi s operačním systémem Windows. Předpokládá se, že na každé stanici je vytvořen administrátorský účet se stejným názvem a heslem.

windows.yml

```
[root@ansible-server group_vars]# cat windows.yml
ansible_ssh_user: admin                                 #specifikace uživatele s oprávněním správce
ansible_ssh_pass: slunickosviti                         #heslo uživatele
ansible_ssh_port: 5986                                  #zvolený port komunikace
ansible_connection: winrm                               #komunikační služba
ansible_winrm_server_cert_validation: ignore            #ignorování validace certifikátů
[root@ansible-server group_vars]#
```

Konfigurace Ansible na systémech Windows

Systémy Windows mohou být spravovány Ansible serverem, pokud bude správně nakonfigurována služba WinRM (Windows Remoting) a bude vytvořen validní self-signed certifikát. Službou bude zajištěna komunikace mezi serverem a spravovanou

stanicí. Samotné spojení probíhá pomocí protokolu HTTPS (Hypertext Transfer Protocol Secure) na portu 5986, který umožňuje šifrovanou komunikaci. Tato nezbytná nastavení mohou být provedena spuštěním skriptu
Configure_Remoting_Ansible_Windows.ps1, který je dostupný na stránkách Ansible [19]. Skript v textové formě je dostupný v příloze A.2.

Nejdůležitějším aspektem pro Ansible na systémech Windows je Powershell. Powershell je implementován již od edice Windows 7, ovšem ve verzi 2.0. Minimální verze pro funkčnost s Ansible je 3.0. Pro zajištění úplné kompatibility a správné funkčnosti se všemi moduly je doporučena instalace Powershell 5.0. Verzi 5.0 je možné nainstalovat na systémech s *.NET Framework 4.6.1*, který je součástí aktualizací systému Windows.

Pokud .NET 4.6.1 není doposud nainstalován, dostupný je na webu zde [20], kde Powershell 5.0 je dostupný na webu [21]. Stručný popis postupu je vyobrazen v příloze A.2.

Konfigurace Ansible serveru pro komunikaci s Linux

Nástroj Ansible byl vytvořen v první řadě za účelem správy linuxových strojů, díky tomu je nastavení komunikace dosti zjednodušeno. Využíván je protokol SSH. Pro zprovoznění komunikace mezi Linux stroji je vyžadována dvojice ssh klíčů, která je složena ze soukromého *id_rsa* A.2.2 a veřejného *id_rsa.pub* A.2.2 klíče. Ansible server disponuje dvojicí těchto klíčů, díky kterým je prokázána identita daného serveru a umožňuje tak inicializaci šifrovaného spojení. K vygenerování klíčů slouží příkaz *ssh-keygen*. Klíče jsou poté uloženy do skrytého souboru *.ssh/* v kořenovém adresáři.

Inicializace spojení bude navázána po přemístění veřejného klíče Ansible serveru na vzdálený linuxový stroj. K tomu bude využit nástroj *ssh-copy-id*, který umožní přemístění a uložení veřejného klíče do souboru *Authorized_keys*, na již zmíněný stroj. Inicializace je zobrazena v příloze A.12.

Tímto je úspěšně dokončeno nastavení komunikace mezi Ansible a Linux stroji. V příloze A.2.2 je vyobrazen test komunikace se stanicí Debian_8 pomocí modulu ping.

3.3 Implementace zadání

V sekci 3.1 bylo specifikováno zadání s instalací rozdílných typů aplikací a různých druhů úkonů, které by měl Ansible splňovat. V následujícím textu budou popsány možnosti, kterými lze tato nastavení docílit spolu s grafickou dokumentací, která bude obsažena v příloze.

3.3.1 Instalace aplikací

Před samotnou instalací je potřeba určit, na jaký typ operačního systému budou aplikace instalovány:

- **Windows**,
- **Linux (typ distribuce)**.

Další podmínkou je určení, dostupnosti aplikací:

- **z repozitáře**,
- **z webu**.

Poslední podmínkou je výběr dostupných modulů ze stránek [12], pro realizaci potřebných požadavků.

Instalace aplikací na Windows

Ansible moduly pro Windows, které jsou dostupné na [22], nabízejí instalaci aplikací pomocí modulu *win_chocolatey*, instalace z repozitáře Chocolatey [23] a modulu *win_package*, který nabízí instalace ze specifikované URL adresy. Modulem je umožněna instalace i odinstalace aplikací:

- **Mozilla Firefox**,
- **Google Chrome**,
- **Putty**,
- **Total Commander**,
- **Wireshark**.

Ukázka struktury modulu pro instalaci Mozilla Firefox.

```
win_chocolatey:                 # název modulu
      name: firefox             # parametr name
      state: present            # parametr state
```

Parametr *name* určuje název aplikace, která bude nainstalována. Parametr *state* definuje stav aplikace, volba *present* určí nainstalování aplikace. Souhrn zbylých parametrů je dostupný na webových stránkách [24]. Modul chocolatey umožňuje instalaci velké škály aplikací, bohužel některé aplikace jsou dostupné pouze ze stránek výrobce. Je tomu tak i při instalaci aplikací od společnosti Microsoft. U takovýchto aplikací je zapotřebí zjistit přesnou URL adresu, na které se nachází *.msi* nebo *.exe* soubor, potřebný k instalaci dané aplikace. Pro instalaci aplikací na Windows z webového nebo lokálního úložiště bude využit modul *win_package*. Nedílnou součástí modulu win_package je klíč aplikace.

Na Obr. A.16 jsou znázorněny klíče instalovaných aplikací. Modulem jsou před instalací zkontrolovány registry, ve kterých se nesmí nacházet shodný klíč. Pokud by došlo ke shodě klíče s klíčem v registru, aplikace by nebyla nainstalována. Bylo by

prokázáno, že aplikace je již nainstalována a nebylo by tedy potřeba ji opětovně instalovat. Hodnoty klíčů instalovaných aplikací jsou dostupné pro:

- **64bit aplikace v:** HKLM:Software\Microsoft\Windows\CurrentVersion\Uninstall,
- **32bit aplikace v:** HKLM:Software\Wow6432Node\Microsoft\Windows\CurrentVersion\Uninstall.

Příkladem může být instalace Miscrosoft Visiual Studio Community pomocí modulu *win_package*.

```
win_package:
    path: "https://download.microsoft.com/download/0/B/C/0BC321A4-013F-479C-84E6-4A2F90B11269/
                vs_community.exe"
    product_id: "{DE064F60-6522-3310-9665-B5E3E78B3638}"
    state: present
```

Parametrem *path* je určeno umístění zdroje .exe nebo .msi souboru potřebného pro instalaci aplikace. Parametr *product_id* specifikuje hodnotu klíče. Parametrem *state* je zajištěna instalace aplikace jako u předešlého modulu. *Win_package* modulem je tedy umožněna instalace jakýchkoliv aplikací dostupných ve formě msi nebo exe souborů. Podrobnější informace o modulu jsou dostupné zde [25]. Výsledný playbook pro instalaci aplikací je dostupný v příloze A.3.1.

Instalace aplikací na Linux

Tak jako pro Windows, tak i pro Linux platí, že ne všechny aplikace jsou spustitelné na daném operačním systému. Pro Linux distribuce nebude uvažována instalace aplikací jako Microsoft Office nebo Visual Studio. Před instalací aplikací na Linux systémy je potřeba, aby bylo určeno, jaké budou použity distribuce. Každá distribuce totiž využívá jiný nástroj pro instalaci aplikací, neboli balíčků.

Například u CentOS a Fedory je využíván stejný formát *.rpm* balíčků, CentOS však využívá nástroj *yum*, Fedora pak nástroj *dnf*. Dle zadání bude využita distribuce Ubuntu a Debian s grafickým prostředím, která využívá stejný formát *.deb* balíčků a k instalaci je využit balíčkovací nástroj *apt*.

Pro instalaci aplikací bude využit modul *apt*, což je obdoba modulu *win_chocolatey* pro Windows. Modul *apt* umožňuje instalaci aplikací, balíčků *.deb* a zároveň i aktualizaci distribucí Ubuntu a Debian viz Kapitola A.3.6. Apt modul pro instalaci aplikace Mozilla Firefox je znázorněn níže.

```
apt:                                    # název modulu
    name: firefox-esr-l10n-cs           # parametr name
    state: present                      # parametr state
```

Výsledný playbook pro instalaci aplikací na Debian a Ubuntu je dostupný v příloze A.3.1.

3.3.2 Informace o stroji

Možnost okamžitého zjištění informací o určitém stroji je velmi výhodná. V tomto textu bude představen modul *debug* pro zobrazení specifických systémových informací pro daný operační systém. Pro zjištění všech informací ze systému Linux poslouží modul *setup*.
Debug modulem je zajištěn srozumitelný výpis, který je pro člověka snadno a rychle čitelný. Pro výběr specifikace informací, které budou ze systému zjištěny, poslouží modul *setup*. Proměnné setup modulu jsou rozdílné jak pro systémy Windows, tak pro Linux. Níže je zobrazena část výpisu pomocí příkazu 2.2.4 pro stanice s Windows.

```
[root@ansible-server ansible]# ansible -i hosts windows -m setup
Windows_7_pc2 | SUCCESS => {
    "ansible_facts": {
        "ansible_architecture": "64-bit",
        "ansible_bios_date": "02.22.2013",
        "ansible_bios_version": "A10",
        "ansible_date_time": {
            "date": "2016-12-10",
            "day": "10",
            "epoch": "1481385605,24229",
            "hour": "16",
            "iso8601": "2016-12-10T15:00:05Z",
            "iso8601_basic": "20161210T160005242286",
            "iso8601_basic_short": "20161210T160005",
            "iso8601_micro": "2016-12-10T15:00:05.242286Z",
            "minute": "00",
            "month": "12",
            "second": "05",
            "time": "16:00:05",
            "tz": "Central Europe Standard Time",
            "tz_offset": "+01:00",
            "weekday": "Saturday",
            "weekday_number": "6",
            "weeknumber": "49",
            "year": "2016"
        },
        "ansible_distribution": "Microsoft Windows 7 Professional ",
        "ansible_distribution_major_version": "6",
        "ansible_distribution_version": "6.1.7601.65536",
```

Pro systém Linux vypadá následovně.

```
[root@ansible-server ansible]# ansible -i hosts linux -m setup
Debian8 | SUCCESS => {
    "ansible_facts": {
        "ansible_all_ipv4_addresses": [
```

```
            "192.168.1.50"
        ],
        "ansible_all_ipv6_addresses": [
            "fe80::20c:29ff:fe9a:cc02"
        ],
        "ansible_architecture": "x86_64",
        "ansible_bios_date": "09/21/2015",
        "ansible_bios_version": "6.00",
        "ansible_cmdline": {
            "BOOT_IMAGE": "/vmlinuz-3.16.0-4-amd64",
            "quiet": true,
            "ro": true,
            "root": "/dev/mapper/Debian--vg-root"
        },
        "ansible_date_time": {
            "date": "2016-12-10",
            "day": "10",
            "epoch": "1481382179",
            "hour": "09",
            "iso8601": "2016-12-10T15:02:59Z",
            "iso8601_basic": "20161210T090259723242",
            "iso8601_basic_short": "20161210T090259",
            "iso8601_micro": "2016-12-10T15:02:59.723379Z",
            "minute": "02",
            "month": "12",
            "second": "59",
            "time": "09:02:59",
            "tz": "CST",
            "tz_offset": "-0600",
            "weekday": "Saturday",
            "weekday_number": "6",
            "weeknumber": "49",
            "year": "2016"
        },
        "ansible_default_ipv4": {
            "address": "192.168.1.50",
            "alias": "eth0",
            "broadcast": "192.168.1.255",
        }
```

Proměnné vložené do uvozovek jsou klíčovými slovy, díky kterým je Ansible schopen dohledat potřebnou informaci. Struktura debug modulu pro výpis informace o typu operačního systému s proměnnou *ansible_os_family* je vyobrazena níže.

```
        - debug: var=ansible_os_family
        - debug: msg="Operacnim systemem je {{ ansible_os_family }}."
```

Výpis modulu debug.

```
TASK [debug] *******************************************************************
ok: [Windows_7_pc2] => {
    "ansible_os_family": "Windows"
}

TASK [debug] *******************************************************************
ok: [Windows_7_pc2] => {
```

```
    "msg": "Operacnim systemem je Windows."
}
```

Playbooky pro získání informací ze stanic využívajících operační systém Windows a Linux spolu se zobrazenými výpisy jsou obsaženy v příloze A.3.2.

3.3.3 Restart a vypnutí stanice

Dalším úkolem zadání bylo umožnění vzdáleného restartu nebo vypnutí stanice. Pro tuto funkci Ansible využívá moduly:

- **Pro Windows:**
  win_reboot,
  win_shell.
- **Pro Linux:**
  shell.

Modul *win_reboot* [26] představuje powershell příkaz pro restart stanice. Je zde využit atribut *reboot_timeout_sec*, kterým je zajištěna časová prodleva pro opětovné navázání spojení se stanicí, když při restartu dojde právě k přerušení spojení mezi Ansiblem a vzdálenou stanicí. Pro restart linuxových stanic je rovnou použit modul *shell* [27], kterým jsou na vzdálenou stanici přenášeny právě shellové příkazy jak pro restart, tak pro vypnutí stroje. Obdobou pro *shell* je modul *win_shell* [28]. Playbooky pro restartování a vypnutí stanic jsou dostupné v příloze A.3.3.

3.3.4 Tvorba uživatelů

Tvorba uživatelů je nedílnou součástí správy sítě. Ansible je schopen vytvářet uživatele jak pro Windows, tak pro Linux. Využit je modul *win_user* [29] a *user* [30]. Oba moduly disponují množstvím nastavení, kterými je například zajištěno přiřazení uživatele do potřebné skupiny, změna nebo nastavení nového hesla. Na systémech Linux je umožněno vytvoření ssh klíče pro daného uživatele. Základní nastavení pro oba moduly je následující.

```
user:                   # nebo win_user:
  name: Petr            # specifikace názvu uživatele
  password: "Admin123"  # vytvoří heslo nového uživatele
  groups: "root"        # určí skupinu do které bude uživatel přiřazen
  state: present        # zajistí vytvoření nového uživatele
```

Použití modulů je znázorněno v příloze A.3.4.

3.3.5 Vytváření složek a souborů

Pro vytvoření souborů a složek je použit modul *win_file* [31] a *file* [32]. Níže je zobrazeno základní použití těchto modulů.

Použití modulu *file*.

```
file:
   path: /home/debian/TextFiles          # určení umístění adresáře
   state: directory                      # vytvoření adresáře

file:
   path: /home/debian/TextFiles/text.txt # určení umístění souboru
   state: touch                          # vytvoření souboru
```

Použití modulu *win_file*.

```
win_file:
   path: path: C:\TextFiles
   state: directory

            file:
   path: C:\TextFiles\text.txt
   state: touch
```

Použití modulů i s modulem *lineinfile* [33] a *win_lineinfile* [34] pro editaci souborů je znázorněno v příloze A.3.5.

3.3.6 Instalace aktualizací

Posledním bodem jsou instalace aktualizací. Aktualizace systému je jednou z nejdůležitějších operací. Díky ní je umožněna stálost a kompatibilita HW se SW a operační systém tak využívá nejnovější informace pro svoji správnou funkci. Pro aktualizaci systému Ansible využívá *win_update* [35] modul pro Windows a *apt* modul pro Linux, o kterém bylo pojednáno v Kapitole 3.3.1.

```
- name: Test system updates
 hosts: windows
 tasks:
    - win_updates:
        category_names:
           - Updates
        state: installed
```

Ve Windows mohou být specifikovány určité typy aktualizací, od bezpečnostních až po aplikační. Pokud nebude specifikována žádná z aktualizací, budou provedeny všechny. Přímé použití modulů pro aktualizaci systému je vyobrazeno v příloze A.3.6.

3.3.7 Odinstalace a mazání

Pro odstranění a odinstalaci aplikací není zapotřebí žádného unikátního modulu, ale postačí pouze zaměnit parametr *present* za *absent*. Tímto je docíleno požadovaného stavu, kterým je samotná odinstalace aplikace, či smazání souboru nebo adresářů, popřípadě uživatelů.

3.3.8 Virtualizační nástroj Vagrant

Nástrojem Ansible je možno provést spousty základní úkony, ale jeho hlavní předností je nasazení nástrojů třetích stran, kterými je například Vagrant. Vagrant [36] je automatizovaný virtualizační nástroj pracující pomocí příkazové řádky nebo terminálu. Byl vyvinut společností *HashiCorp* pro usnadnění a urychlení práce s virtuálními stroji. Vagrant je psán v programovacím jazyce Ruby a je schopen spolupracovat s virtualizačními nástroji *VMware Player*, *VirtualBox*, *Hyper-V*.

Jednoduše řečeno je Vagrant nadstavbový systém pro ovládání výše zmíněných virtualizačních systémů. Nejdůležitější součástí Vagrantu je *Vagrantfile*, což je konfigurační soubor. V konfiguračním souboru jsou obsaženy informace o virtuálním stroji, jako je IP adresa, název stroje, a další.
Níže je vyobrazen alespoň částečný obsah souboru *Vagrantfile*, který byl vygenerován následujícím příkazem.

```
vagrant init centos/7
```

Názvem *centos/7* je určen typ distribuce spolu s jeho verzí. V tomto případě bude vytvořen virtuální stroj s CentOS 7.

```
Vagrant.configure("2") do |config|
  config.vm.box = "centos/7"
  # config.vm.box_check_update = false
  # config.vm.network "forwarded_port", guest: 80, host: 8080
  # config.vm.network "private_network", ip: "192.168.33.10"
  # config.vm.network "public_network"
  # config.vm.synced_folder "../data", "/vagrant_data"
  # config.vm.provider "virtualbox" do |vb|
  #   vb.gui = true
  #   vb.memory = "1024"
  # end
end
```

Při vygenerování výchozího souboru *Vagrantfile* je 90 % nastavení zakomentovaných. Konfigurace obsahuje například nastavení IP adres, portů, velikosti RAM, výběr poskytovatele virtualizace (VMware, VirtualBox, atd.) a dalších. Příkaz

```
config.vm.box = "centos/7"
```

obstarává stažení boxu se stanicí s CentOS 7, ze stránek *atlas.hashicorp.com* [37]. Stránky jsou výchozím úložištěm boxů se systémy Windows a Linux.

Po vygenerování a nastavení potřebných parametrů v souboru *Vagrantfile* je nutné virtuální stroj spustit příkazem *vagrant up*. Při prvotním spuštění dojde nejprve ke stažení boxu obsahujícího virtuální stroj, kdy po úspěšném stažení dojde k okamžitému spuštění virtuálního stroje. Příkaz *vagrant status* objasní aktuální stav virtuálního stroje. Pro konečný přístup do virtuálního stroje z příkazové řádky je využit příkaz *vagrant ssh*, který zprostředkuje komunikaci z fyzického stroje do nově vytvořeného virtuálního stroje.

Potřebné kroky spolu se softwarem a playbookem, zajišťujícím hromadnou instalaci pomocí nástroje Ansible, jsou dostupné v příloze A.3.7.

4 TVORBA WEBOVÉ APLIKACE

V předchozí kapitole byla navržena, implementována a otestována správa počítačové sítě pomocí nástroje Ansible. Bylo ověřeno, že nástroj Ansible je dostatečným a nenáročným nástrojem pro obsluhu i rozsáhlejších sítí, obsahujících nejen operační systém Linux, ale i Windows. Systémový administrátor tak dokáže aplikovat sady úkonů, a to od jednoduchých (vytváření složek, přesuny souborů) až po složitější (instalace a konfigurace aplikací, nastavování oprávnění).

I přesto, že nástroj Ansible není nijak náročný na pochopení a obsluhu, stále však nutí administrátora používat terminálové okno pro vytváření playbooků, rolí a spouštění příkazů. Tato obsluha se po určité době stane časově náročnou a neefektivní. Dále je administrátor odkázán na obsluhu Ansible pouze z interní sítě, kterou však může ulehčit například připojení přes VPN (Virtual Private Network) z jiné sítě. Tento způsob zahrnuje zbytečné množství kroků jako je již zmíněné přihlašování na VPN, následné přihlášení na server, atd.

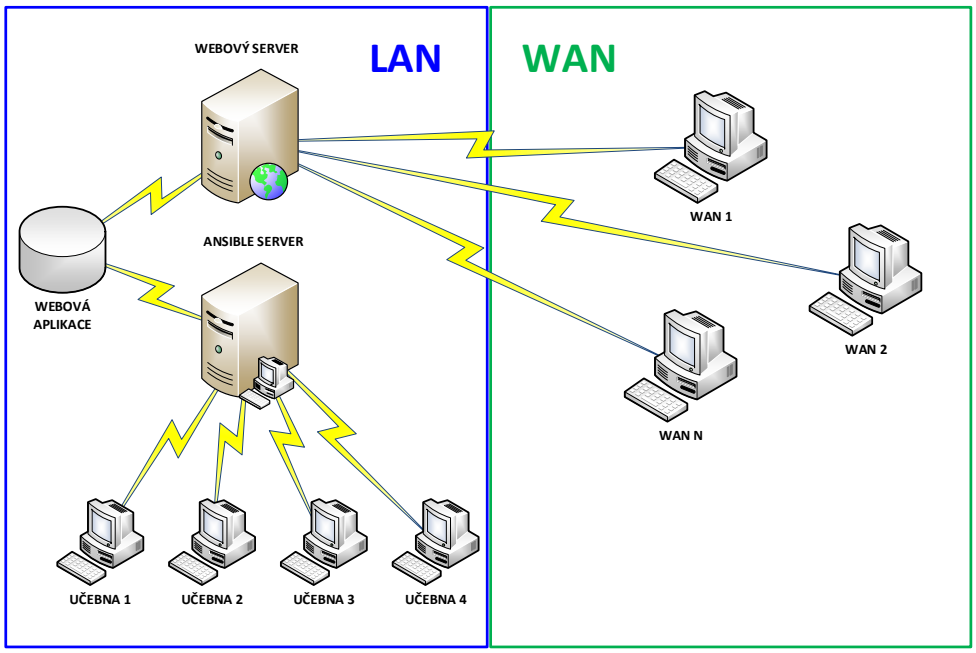
Tyto problémy lze vyřešit pomocí rozhraní, které by bylo přístupné z jakékoliv sítě připojené k Internetu a umožňovalo by obsluhu pomocí GUI (Graphical User Interface) rozhraní. Výsledkem by mohlo být webové rozhraní s grafickými ovládacími prvky. Ansible by tak měl grafickou nadstavbu pro ovládání a obsluhu a byl by dostupný z jakéhokoliv webového prohlížeče (Google Chrome, Mozilla Firefox, Internet Explorer) pod určitou URL (Uniform Resource Locator) adresou. Komunikace uživatele s webovým serverem by probíhala pomocí HTTP nebo šifrované HTTPS komunikace.

4.1 Návrh architektury

Pro vytvoření funkční webové aplikace je zapotřebí navrhnout architekturu, díky které bude webová aplikace fungovat dle zadaných požadavků. Z Obr. 4.1 je patrná možná komunikace klienta se serverem, nacházejícího se ve veřejné síti, který preferuje obsluhu nástroje Ansible pomocí webového rozhraní. Jak bylo zmíněno v Kapitole 4, aby daný uživatel mohl přistoupit na webovou stránku, obsluhující Ansible bude muset zadat buď veřejnou IP adresu nebo DNS (Domain Name System) název webového serveru, který bude zprostředkovávat komunikaci s webovou aplikací.

Webová aplikace je nejdůležitějším článkem celé architektury, jelikož bude mít na starosti vyhodnocování a zpracování dotazů od klientů vůči nástroji Ansible. Aplikace by tak měla dle typu dotazů umožňovat například spuštění vybraného playbooku na specifikovanou skupinu hostů nebo vytvářet, editovat či mazat informace obsažené v daných souborech s playbooky. Dále by měla umožňovat ověřování

při přístupu do webového rozhraní, a to jak pomocí loginu, tak hesla. Také by měla poskytovat výpisy z právě provedených úkonů spolu s dalšími funkcemi, které jsou pro Ansible nezbytné.

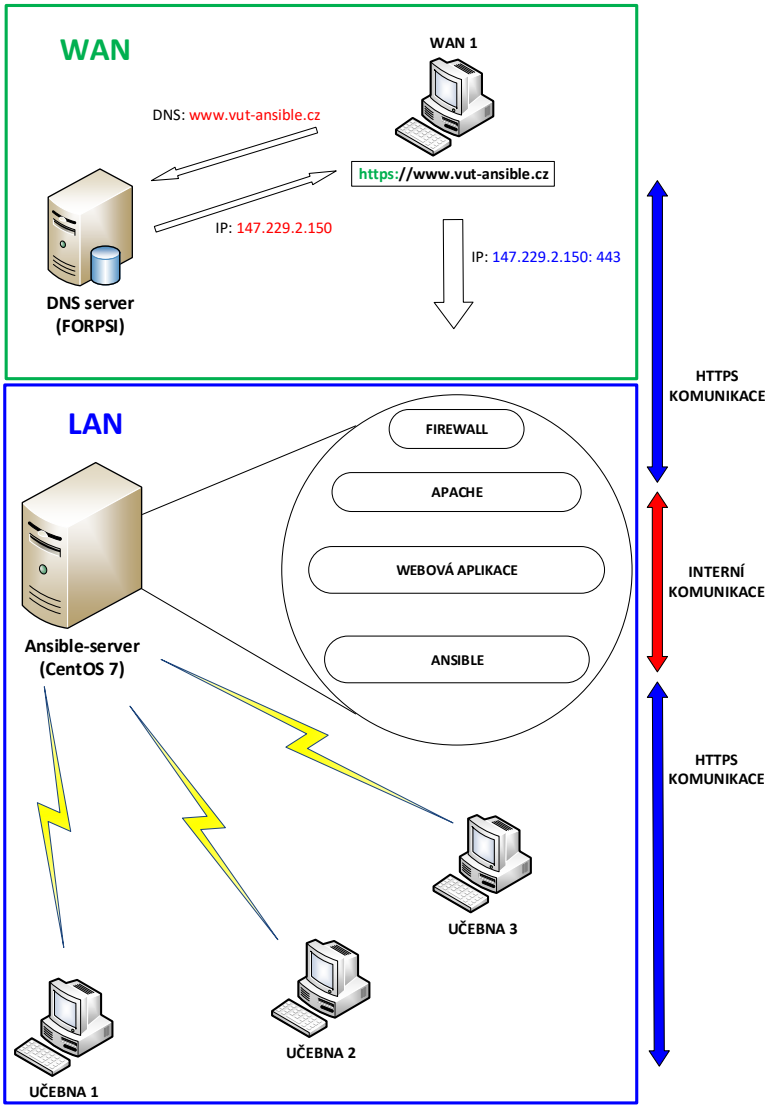


Obr. 4.1: Architektura komunikace klienta s webovou aplikací.

Obr. 4.2 je důkladněji vyobrazena komunikace webové aplikace s prohlížečem vzdáleného klienta. Jak je vidět, tak pro zprovoznění webové aplikace obsluhující Ansible nepostačí vytvořit pouhou aplikaci, ale je zapotřebí součinnosti několika dalších nástrojů, které budou popsány v následujícím textu.

4.2 Potřebné nástroje

Aby byla aplikace dostupná z veřejné sítě, je potřeba vlastnit doménu. Doménou se rozumí doménový název, jako například ***www.vut-ansible.cz*** , ke kterému bude přiřazena IP adresa směřující na webový server (v tomto případě Apache). Tento webový server pak dokáže směrovat požadavky přímo do webové aplikace, která je spuštěna na hostitelském systému. Samotná aplikace pak již obstarává přijímané dotazy od klienta a pomocí metod spouští nebo vykonává zadané úkony. Níže budou vyjmenovány a stručně popsány všechny potřebné nástroje spolu s jejich konfigurací. Nástroje budou rozděleny do dvou skupin. První skupina s názvem *vývoj* bude určena pro samotný vývoj aplikace. Druhá skupina s názvem *provoz* bude obsahovat



Obr. 4.2: Podrobný popis komunikace klienta s webovou aplikací.

všechny nástroje potřebné ke zdárné funkčnosti a běhu webové aplikace.

4.2.1 Vývoj

Před započetím vývoje webové aplikace je zapotřebí, aby bylo objasněno několik základních požadavků. V dnešní době existuje řada programovacích jazyků, které jsou rozděleny do několika odvětví, a to programovací jazyky pro mobilní aplikace, desktopové aplikace, webové aplikace a jiné. V tomto případě bude řeč o programovacím jazyku, který je určen právě pro tvorbu webových aplikací. Nejznámějšími jazyky pro psaní webových aplikací jsou *PHP* a *Java* nebo spíše jejich frameworky,

jako je *Symfony* pro PHP a *Spring* pro JAVU. Nám bude objasněna tvorba webové aplikace pomocí programovacího jazyku *Java*, a to s využitím frameworku *Spring*.

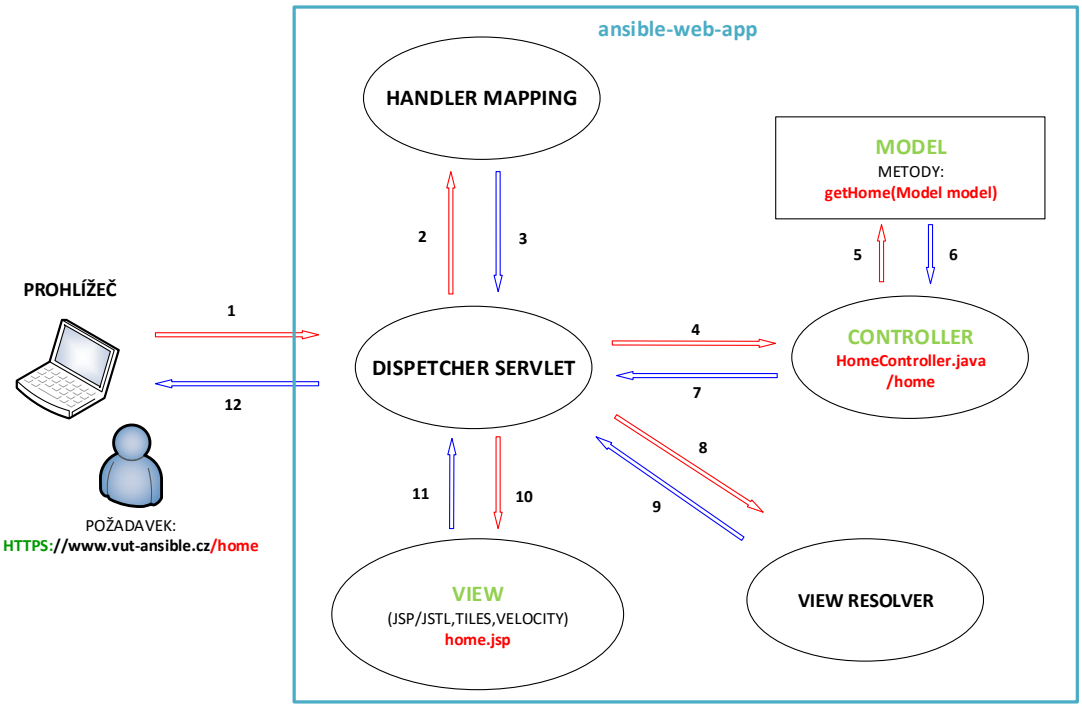
Pro vývoj aplikace využívající framework Spring bude použito vývojové prostředí *STS* (Spring Tool Suite)[38], které využívá základu Eclipse a je přímo určeno pro práci se Spring frameworkem. V poslední řadě bude představen nástroj *Maven*, který je další nedílnou součástí pro usnadnění práce.

Spring - Web MVC Framework

Spring Web MVC framework využívá architekturu *MVC* (*Model-View-Controller*), díky které je poskytnut komplexní programovací a konfigurační model pro moderní podnikové aplikace založené na jazyce Java. Architektura MVC má za následek oddělení aspektů aplikace, kterými jsou *vstupní logika*, *obchodní logika* a *uživatelská logika*. Mezi těmito prvky je však vytvořena volná vazba. Každá část modelu MVC má na starosti specifický úkol.

- **Model:** Java objekt, který udržuje informace o datech aplikace, např. jaké má objekt uživatel jméno a heslo. Jsou tak volány metody, které vrací data obsažená na požadované stránce.
- **View:** Opět Java objekt, který uvnitř uchovává informaci o grafickém rozložení webové stránky spolu s daty obsaženými v modelu. Výstupem jsou data formátu *JSP* (JavaServer Pages), ve kterých jsou uložena požadovaná data s grafickým vzhledem webové stránky.
- **Controller:** Nejdůležitější část modelu MVC. Controller má na starosti dle požadované cesty v URL adrese např. */home* vyhledat požadovaná data, která volaná stránka obsahuje spolu s určením grafické šablony, díky které volaná stránka dostane svůj výsledný vzhled.

Pomocí Obr. 4.3 je podrobně popsána komunikace uvnitř "springové"aplikace.



Obr. 4.3: Architektura Spring Web MVC Framework.

Na Obr. 4.3 je vyobrazen požadavek na stránku https://vut-ansible.cz/home, který byl poslán z klientova webového prohlížeče pomocí zabezpečeného protokolu https. Požadavek na stránku byl poslán pomocí GET metody. Po krátké výměně informací klientského počítače s DNS serverem dorazí požadavek na server, kde běží webová aplikace. Aplikace se skládá navíc z:

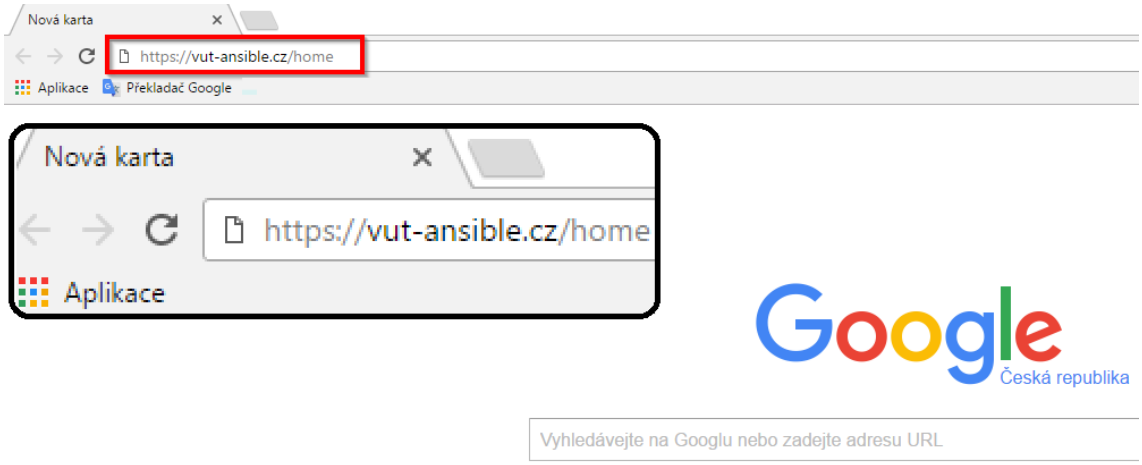
- **Dispetcher Servlet:** Hlavní řídící bod aplikace. Zajišťuje komunikaci uvnitř aplikace a přijímá a odesílá HTTP dotazy a odpovědi.
- **Handler Mapping:** Předává informace o tom, kde se nachází požadovaný controller.
- **View Resolver:** Předává informace o tom, kde se nachází požadovaný grafický vzhled stránky.

Níže je v bodech popsána podrobná komunikace procházející webovou aplikací.

1. Žádám o načtení stránky vut-ansible.cz/**home** pomocí HTTP žádosti metodou GET, Obr. 4.4.
2. Kde se nachází controller pro stránku **/home** volaný pomocí HTTP žádosti metodou GET?

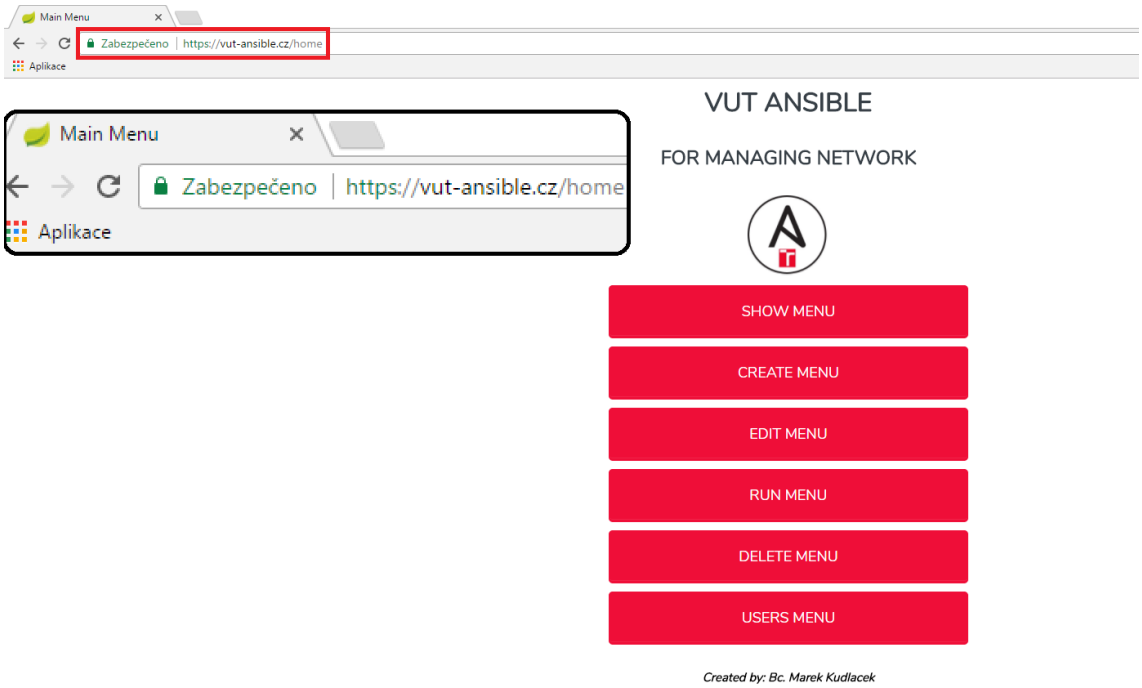
3. Dotaz na stránku **/home** pomocí metody GET zajišťuje controller **HomeController.java**.
4. Zpřístupni všechna data dostupná v controlleru **HomeController.java**.
5. Zpřístupni všechna data obsažená v metodě **getHome()**, která je obsažena v controlleru **HomeController.java**.
6. Metoda vrací tyto data spolu s názvem grafické šablony **home.jsp**.
7. V **HomeController.java** byla obsažena tato data spolu s názvem šablony **home.jsp**.
8. Kde se nachází šablona **home.jsp**?
9. Šablona se nachází v **WEB-INF/jsp**.
10. Zpřístupni data obsažená v šabloně **home.jsp**.
11. Zde jsou data šablony **home.jsp**.
12. Zde je stránka s grafickým vzhledem šablony **home.jsp** a daty z metody **getHome()**, Obr. 4.5.

Z předešlých kroků je patrná výhoda, Spring frameworku, kdy se programátor může zaměřit pouze na psaní čistého kódu a pracuje hlavně s prvky Model, View a Controller. Práci Dispetcher Servlet, Handler Mapping a View Resolver, přebírá Spring a stará se tak o potřebné návaznosti, které programátor vyřeší pomocí anotací (*Annotations*). Kroky potřebné pro práci s vývojovým prostředím *STS* budou představeny v příloze 4.2.1.



Obr. 4.4: Požadavek na stránku vut-ansible.cz/home.

V příloze A.4.1 je znázorněna praktická ukázka všech návazností od zaslání požadavku na stránku, průchod modelem MVC až po výsledný vzhled požadované stránky.



Obr. 4.5: Odpověď od serveru na stránku vut-ansible.cz/home.

Aby mohla být vyvíjena webová aplikace pomocí Spring frameworku, je potřeba doinstalovat a dokonfigurovat několik potřebných aplikací jako *JDK* (Java Development Kit) 4.2.1, *Maven* 4.2.1 a *STS* (Spring Tool Suit) 4.2.1, které budou popsány v následující části textu. Seznam všech potřebných a doporučených aplikací je také dostupný na stránkách *spring.io/docs* [39].

JDK (Java Development Kit)

Je využívána buď jako běhové prostředí (bude využito v Kapitole *Provoz* 4.2.2), nebo jako vývojové prostředí, které bude využito pro vývoj webové aplikace. Java je rozdělena do několika verzí.

- **Java SE (Standard Edition)** – základní verze Javy,
  Java SE Development Kit – hlavní nástroj pro vývoj programů v jazyce Java, jeho součástí je i JRE,
  Java SE Runtime Environment – obsahuje sadu knihoven a programů, které umožňují spouštět programy psané v Javě,
- **Java EE (Enterprise Edition)** – rošířená verze SE využívaná pro vývoj aplikací v průmyslu,
- **Java ME (Micro edition)** – Java pro vývoj aplikací pro mobilní zařízení.

V tomto případě bude potřeba nainstalovat *Java SE Development Kit* ve verzi 1.8.0, který je zdarma ke stažení na stránkách www.oracle.com [40]. Postup instalací je objasněn v příloze A.4.1.

Maven

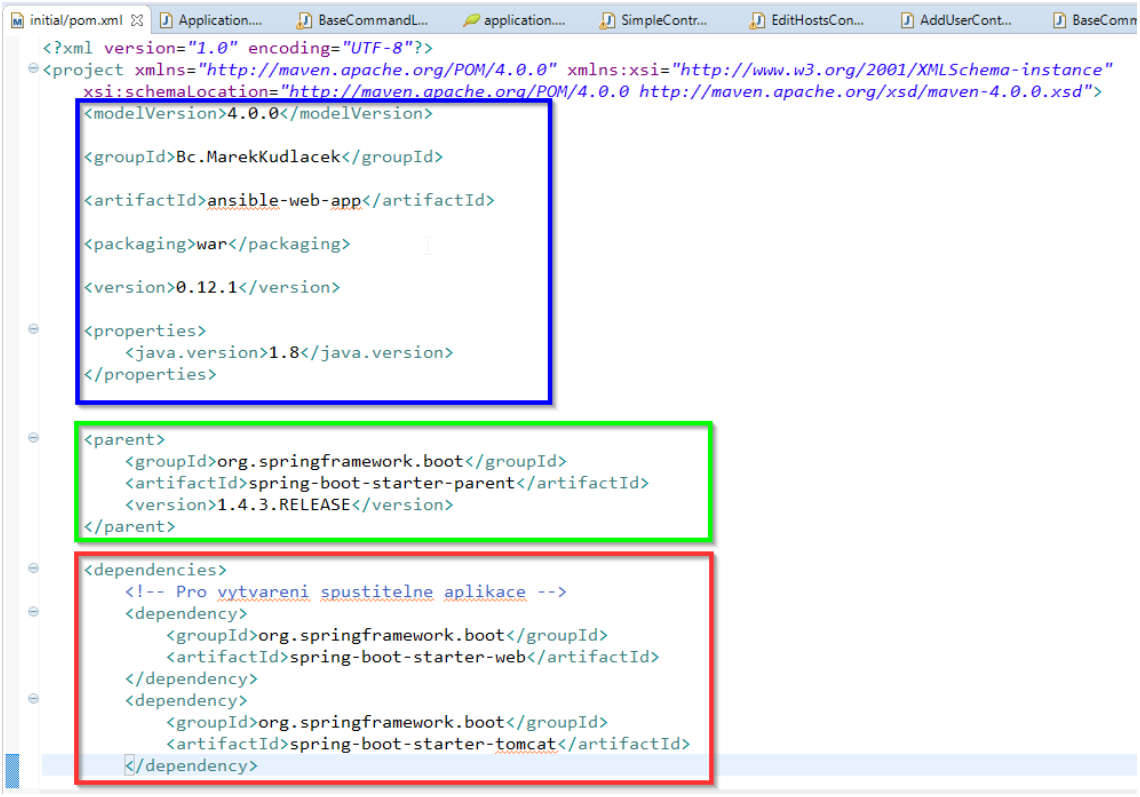
Apache Maven, nástroj řídící automatizovanou správu sestavení (buildů) aplikací. Pomáhá usnadnit práci při sestavení aplikace a režie s tím spojené. Ke své práci využívá projektově objektového modelu *POM* (Project-Object-Model) spolu s pluginy, které zprostředkují programátorovi např. všechny potřebné knihovny, které při použití určité třídy nebo metody bude vyžadovat. Je tak ušetřen čas, který by programátor musel vynaložit při hledání potřebných návazností pro zdárné sestavení kódu. Maven do projektu vnáší ucelené postupy a pravidla, které se již v praxi osvědčily. Odděluje zdrojové kódy tříd, unit testů a ostatních souborů, jako jsou obrázky, properties, atd. Adresáře obsahující oddělené soubory:

- **src/main/java** – zdrojové kódy tříd,
- **src/test/java** – unit testy,
- **src/main/resources** – obrázky, šablony css, vlastnosti.

Pro správný chod Mavenu je tedy zapotřebí tzv. *pom.xml* soubor, Obr. 4.6, který je součástí projektu. Uvnitř souboru se nachází konfigurační informace (elementy), díky kterým Maven dohledává návaznosti, které jsou nezbytné pro správnou funkčnost dané aplikace.

Maven obsahuje informace, které mohou být obsaženy v konfiguračním souboru. Z Obr. 4.6 budou objasněny ty nejzákladnější elementy, které jsou potřeba pro úspěšné sestavení projektu. Podrobné informace o elementech používaných v Mavenu jsou dostupné na stránkách *maven.apache.org* [41].

- **<project>** kořenový element Mavenu, tzv. popisovač (descriptor). Vkládají se do něj všechna nezbytná data,
- **<modelVersion>** určuje verzi použitého popisovače pro pom soubor,
- **<groupID>** název skupiny nebo organizace, které projekt patří,
- **<artifactID>** název projektu,
- **<packaging>** typ formátu do kterého bude projekt zabalen (.jar, .war),
- **<version>** číslo verze projektu,
- **<java.version>** číslo verze JDK, které je použito pro vývoj aplikace,
- **<parent>** určení umístění rodičovského projektu, ze kterého bude projekt dědit informace. V tomto případě je to již zmiňovaný Spring framework.



Obr. 4.6: Konfigurační soubor pro Maven *pom.xml*.

- **<dependencies>** v tomto elementu se určují závislosti potřebné pro chod projektu.

Po sestavení projektu pomocí příkazu níže, bude předvedeno v příloze A.4.1, je vrácen název aplikace složený z *<artifactID>* a *<version>*, kde formát souboru bude podle elementu *<packiging>*.

```
mvn package
```

Výsledek pak bude vypadat následovně:

- *ansible-web-app-0.12.1.war*.

Tímto vznikne spustitelná aplikace, která bude poté nasazena na server, kde bude vykonávat zadané požadavky.

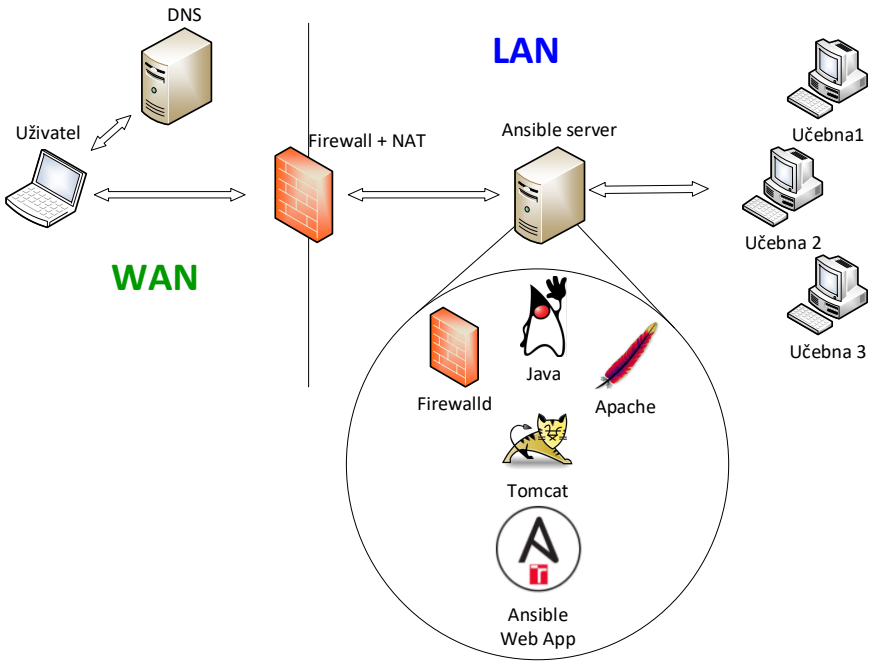
STS (Spring Tool Suit)

Spring Tool Suit je vývojovým prostředím, které bylo vyvinuto právě pro práci s frameworkem Spring. STS je přetvořeným vývojovým prostředím Eclipse a přebírá tak od něj všechny známé funkce jako vytváření projektů, debagování, deploy a další.

Uvnitř Spring Tool Suit je zahrnuta kombinace podporující jazyk Java, Spring framework, Pivotal tc Server, Git, Maven, Gradle. Je to balík obsahující ty nejpotřebnější komponenty pro vývoj webových aplikací. STS ocení převážně uživatelé, kteří se již setkali s vývojovým prostředím Eclipse. Pokud je rozhodnuto pro vývoj ve vývojovém prostředí STS, postačí stáhnout zabalený soubor obsahující spustitelnou aplikaci *STS.exe*, jak je ukázáno v příloze A.4.1.

4.2.2 Provoz

V předešlé Kapitole 4.2.1 byly probrány nástroje potřebné pro vývoj webové aplikace. V této části textu budou probrány nástroje, které jsou potřebné pro provoz aplikace na serveru. Aby aplikace splňovala všechny požadavky pro komunikaci, je důležité správně nakonfigurovat nezbytné komponenty, jako je např. firewall, DNS (Domain Name System), webový server a další. Na Obr. 4.7 jsou vyobrazeny nástroje použité při komunikaci uživatele s webovou aplikací.



Obr. 4.7: Nástroje potřebné pro provoz webové aplikace.

JRE (Java Runtime Environment)

V Kapitole 4.2.1 byly představeni zástupci programovacího jazyka Java. Byla zde představena i *Java Runtime Environment*, která je využita právě pro spouštění programů a aplikací napsaných v jazyce Java. Aby bylo možné spustit vytvořený *ansible-web-app-0.12.2.war* soubor, tak je do systému doinstalována *java-1.8.0-openjdk*, která zajistí možné spuštění aplikace a její správnou funkčnost.

Pro zdárné nainstalování rpm balíčku na CentOS7, postačí použití příkazu:

```
yum install java-1.8.0-openjdk
```

Následně již nic nebrání spuštění aplikace.

```
nohup java -jar ansible-web-app-0.12.1.war > /root/web-app/log.txt 2>&1
```

Dodatečné informace a význam použitých příkazů jsou obsaženy v příloze A.4.2.

Apache

V předešlé části textu 4.2.2 byla aplikace spuštěna, ale stále je dostupná pouze na adrese *localhost:8080*. Aplikace by ovšem měla být ve výsledku dostupná na portech 80 a 443, což jsou porty pro komunikaci pomocí aplikačních protokolů *HTTP* a *HTTPS*. Aby aplikace byla přístupná pomocí těchto dvou protokolů, je zapotřebí nainstalovat webový server. V této diplomové práci bude probrána a názorně předvedena práce s webovým serverem *Apache* ve verzi 2.4.6.

Apache je jedním z nejrozšířenějších webových serverů po celém světě. Spadá do rodiny nástrojů s otevřeným kódem (open source). Je podporován na platformách jako je Microsoft Windows, BSD a Linux. Obsahuje nesčetné množství funkcí, které mohou být rozšířeny o takzvané *moduly*. Každý modul má na starosti rozšiřující funkci, jako *mod PHP* (modul pro práci se skriptovacím jazykem PHP), *mod rewrite* (modul pro přesměrování URL adres) a např. *mod ssl*, který bude použit v následující části textu. Tak jako Maven i Apache je vytvořen společností *Apache Software Foundation*.

Jeho funkcí je uživatelům zprostředkovat pomocí *URL* adres hledaná data a informace obsažené na webových stránkách. Apache slouží jako prostředník mezi uživatelem a aplikací, který zodpovídá požadované dotazy. Ke své funkci využívá aplikačního protokolu *HTTP*.

- **HTTP (Hyper Text Transfer Protocol)**
  Metody dotazů
  HTTP protokol pro přenos dotazů využívá metody, kterými specifikuje požadavky na zdroje serveru. Nejvíce používanými jsou:

  - **GET**
    Slouží k vyzvednutí objektu (html soubor, obrázek, atd.), odpověď se ukládá do cache a doprovází ji redundantní počet hlaviček specifikujících informace o souboru.
  - **POST**
    Zajišťuje přenos informací od uživatele na server. Umožňuje tak např. zaslat přihlašovací jméno a heslo pomocí webových formulářů.
  - **DELETE**
    Dokáže smazat vybraný objekt na serveru.

- **OPTIONS**
  Zjístí informace o možnostech spojení.

Klientské hlavičky

V každém dotazu jsou ukryty hlavičky, ve kterých se nacházejí požadované informace od serveru, jsou jimi:

- **Host**
  Doménové jméno virtuálního hostitele. (Na jednom serveru tak může být pod jednou IP adresou skryto několik virtuálních hostitelů s různými doménami.)
- **User-Agent**
  Řetězec identifikující klienta.
- **From**
  Adresa uživatele zodpovědného za tento požadavek.
- **If-Modified-Since**
  Žádost o načtení aktuálních dat dokumentu, pouze pokud byly od tohoto data modifikovány.

Serverové hlavičky

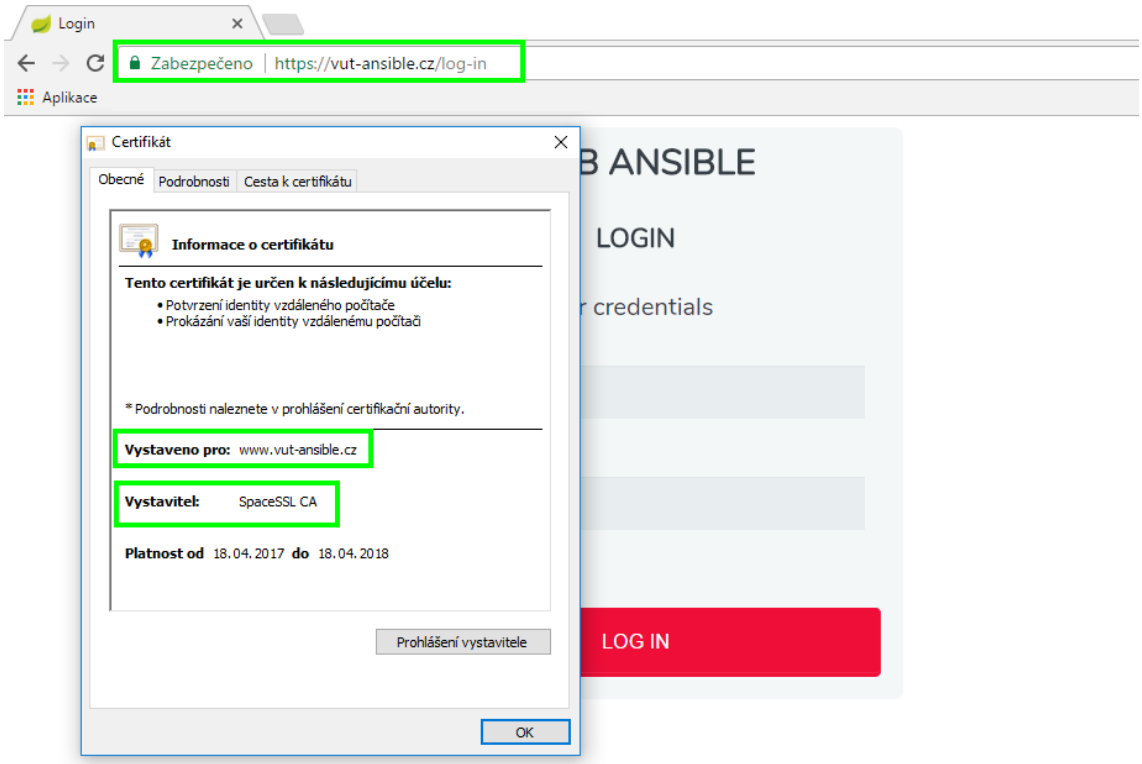
Pokud server odesílá odpovědi na dotazy, tak k těmto účelům využívá následující hlavičky:

- **Content***
  Popisuje obsah (tělo) požadavku. Např. *Content-Type* typ dokumentu.
- **Date**
  Současné datum odpovědi.
- **Expires**
  Doba do vypršení platnosti dokumentu.
- **Server**
  Verze a typ serveru.

- **HTTPS (Hyper Text Transfer Protocol Secure)**
  Obdoba HTTP protokolu umožňující šifrovanou komunikaci. Ke komunikaci využívá stejně jako HTTP spojení pomocí transportního protokolu TCP, využívajícím port 443. Šifrování je zajištěno protokolem *TLS* (Transport Layer Security), kterým je zajištěna konzistence a validita zpráv. HTTPS využívá metodu asymetrické kryptografie. Ta spočívá ve vygenerování dvou klíčů. Prvním je *privátní klíč* (Privare Key), který nesmí být nikdy prozrazen a musí se nacházet na serveru, kde se budou nacházet požadovaná data. Druhým je *veřejný klíč* (Public Key), který je poskytován uživatelům a následně je ověřo-

ván vůči soukromému klíči. Klíče jsou tzv. *certifikáty*, které mohou být dvojího typu:

- **Podepsaný sám sebou (self-signed)**
  Certifikáty vygenerované svépomocí, vhodné pro interní účely v místní síti, nikoliv v Internetu.
- **Podepsaný certifikační autoritou (CA)**
  Stejný jako v předešlé části, avšak je podepsán důvěryhodnou certifikační autoritou (*Geo Trust*, *COMODO*, *Thawte*) a další. Certifikát je určen k použití ve veřejné síti, kde je kladen důraz na ověření pravosti certifikátu certifikační autoritou. Na Obr. 4.8 je znázorněn ověřený certifikát pro doménu *www.vut-ansible.cz* od certifikační autority *SpaceSSL CA*.



Obr. 4.8: Ověřený certifikát vystavený pro doménu www.vut-ansible.cz.

Aby Apache odpovídal na dotazy jak na portu 80 tak i 443, tak je nutné provést správnou konfiguraci. Konfigurace je prováděna pomocí *direktiv*. Tyto direktivy specifikují informace, kterými je webový server ovládán. Pro seznámení bude část z nich představena. Direktivy jsou děleny na *párové* a *nepárové*. Použití direktiv je však závislé na použitém *kontextu*.

- **Direktivy nepárové**

  ServerRoot [**cesta k souboru s konfigurací**] – určuje domovský adresář webového serveru.

  ServerName [**plný doménový název serveru nebo IP adresa:**[**port**]] – identifikuje server na kterém je Apache spuštěn. Identifikace probíhá pomocí doménového jména nebo IP adresy serveru.

  DocumentRoot [**cesta obsahující html soubory**] – jsou zde obsaženy soubory potřebné k prezentaci webové stránky.

  Listen [**IP adresa:** [**port**]] – určí serveru na které IP adrese a portu má očekávat komunikaci.

  ErrorLog [**cesta k souboru**] – zajistí ukládání chybových logů do specifického umístění.

  KeepAlive [**on/off**] – povolí nebo vypne perzistentní spojení serveru. Umožní přenos několika požadavků přes jedno TCP spojení.

  ServerAlias [**jméno aliasu**] – přiřazuje alternativní názvy k hlavnímu ServerName, využíváno hlavně v kontextu VirtualHost.

  Timeout [**čas v sekundách**] – čas po který je udrženo spojení se serverem. Ve výchozím nastavení 300 sekund.

  ProxyPass [[**URL na místním serveru**] [**vzdálená URL**]] – využívá se pro mapování vzdálených URL požadavků na URL adresy obsažené na serveru.

  RewriteEngine [**on/off**] – po zapnutí umožňuje přesměrování mezi virtuálnímy hosty.

- **Direktivy párové**

  <VirtualHost [**host**]**>**

Zprostředkuje možnost dostupnosti několika hostitelů, přístupných pod jediným webovým serverem. Direktiva obsahuje pouze informace, které se týkají konkrétního virtuálního hostitele. Jednotliví hostitelé jsou rozlišováni podle doménových názvů za pomoci direktivy *ServerName*. Tento název je shodný s hlavičkou *Host* zaslanou v požadavku protokolu HTTP/1.1. Touto cestou tak server dokáže odlišit jednotlivé virtuální hostitele obsažené na serveru.

  </VirtualHost>

  <Directory [**parametry**]**>**

Specifikuje vyhovující parametry pro adresáře a podadresáře souborového systému. Musí být použita absolutní cesta k adresáři.

  </Directory>

  <Files [**parametry**]**>**

Má podobné vlastnosti jako předešlá direktiva, zahrnuje však soubory.

</Files>

<IfModule [modul]>

Využíván v souladu s použitým modulem.

</IfModule>

- **Kontexty**

  Určují kde a jak mohou být direktivy použity.

  Directory – využíván u párových direktiv (Directory, Files, Proxy a Location).

  Virtual Host – obsahuje nastavení tzv. sekundárních webových serverů, využívajících společné systémové prostředky Apache.

  .htaccess – podobný jako kontext *Directory* s tím, že uživatelé kteří nemohou zasahovat do konfigurace Apache mohou alespoň specifikovat direktivy pro daného hostitele.

  Server Config – platí pro konfiguraci obsaženou přímo uvnitř konfiguračního souboru *httpd.conf*.

Tímto bylo završeno představení funkcí a vlastností webového serveru Apache. Praktické konfigurace webového serveru s módem SSL (mod_ssl) jsou probrány v příloze A.4.2.

Tomcat

Tomcat podobně jako Apache spadá do rodiny webových serverů. Oproti Apache serveru využívá *Java Servlet Container*. Servlet Container zodpovídá za dynamické generování webových stránek na straně serveru. Spolehlivě pracuje s technologiemi jako jsou *Java Servlet*, *JavaServer Pages* (využito ve Spring aplikaci) nebo *Java EL* a *WebSocket*.

Tomcat poskytuje prostředí Javy pro webové servery, kde může být spuštěn kód, který byl napsán v jazyce Java. V diplomové práci není použit Tomcat jako takový, ale je vyžíván tzv. *Embedded Tomcat*, který je součástí aplikace psané pomocí frameworku *Spring*. Do aplikace je přidán pomocí závislostí nástrojem *Maven* a je tedy součástí konfiguračního souboru *pom.xml*, jak je znázorněno níže.

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-tomcat</artifactId>
</dependency>
```

Obr. 4.9: Tomcat přidaný do Spring aplikace pomocí Maven závislosti.

Firewall a NAT

V následující části textu bude ve zkratce objasněn význam *Firewallu* a služby *NAT* (Network Address Trasnlation). Informace jsou čerpány ze zdrojů [42] a [43]. Firewall je nedílnou součástí každého systému a jeho úkolem je bránit uživatele před nežádoucím provozem ze sítě. Firewall propouští komunikaci za předpokladu, že byla splněna určitá pravidla. V současné době jsou firewally rozděleny do dvou základních druhů.

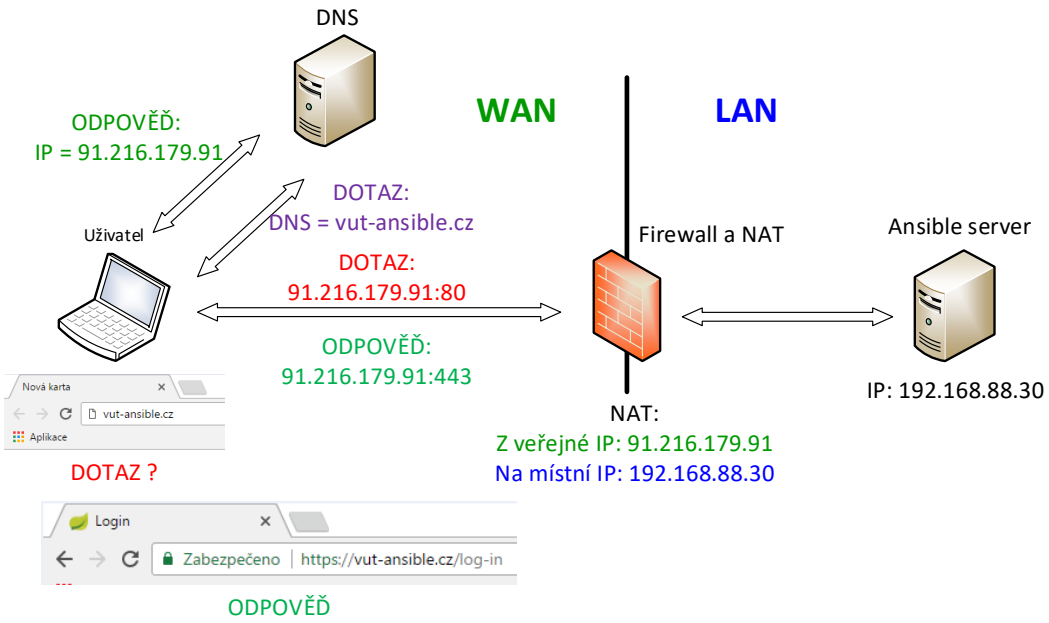
- **Softwarové** – levné, více dostupné, méně výkonné (iptables, firewalld, ipchains),
- **Hardwarové** – drahé, dostupné, výkonné (CISCO, HP, Jupiter).

Dále mohou být členěny podle činnosti.

- **Paketový** – komunikace rozlišena na základě IP adres,
- **Stavový** – přidává pravidla podle stavu komunikace,
- **Aplikační** – dokáže pracovat i s aplikačními protokoly.

V diplomové práci jsou využity oba druhy firewallů. Hardwarový firewall, od společnosti CISCO, je využit jako hraniční firewall, přes který probíhá veškerá komunikace do lokální sítě, jak je znázorněno na Obr. 4.7. Zároveň je na něm zprovozněna i překladová služba *NAT*, která zprostředkuje překlad místních adres na veřejné a naopak. Objasnění funkce NAT je znázorněno na Obr. 4.10.

Softwarový firewall *firewalld* je využit na Ansible serveru, kterým je zabezpečena komunikace pomocí protokolu *HTTP* a *HTTPS*. Tyto aplikační protokoly jsou určeny pro přístup do webové aplikace.



Obr. 4.10: Zjednodušené objasnění funkce NAT.

Praktické konfigurace firewallu a služby NAT pro zpřístupnění webové aplikace z veřejné sítě jsou probrány v příloze A.4.2.

DNS

Každý webový zdroj je dostupný pod určitým názvem ať je to *www.seznam.cz*, *www.facebook.com* nebo *www.google.com*. Internet pro své směrování využívá IP adresy, ty jsou však pro člověka těžko zapamatovatelné. Proto je využíván systém doménových názvů *DNS*, který umožňuje překlad doménových názvů na IP adresy. Je tedy vytvořen pár *doménový název* a *IP adresa*. Například:

- **Doménový název:** www.seznam.cz – **IP:** 77.75.77.39,
- **Doménový název:** www.facebook.com – **IP:** 31.13.70.36,

- **Doménový název:** www.google.com – **IP:** 172.217.20.196.

Zjištění IP adresy z doménového názvu je možné nástrojem *ping*.

```
ping www.seznam.cz
Pinging www.seznam.cz [77.75.77.39] with 32 bytes of data:
Reply from 77.75.77.39: bytes=32 time=13ms TTL=247
Reply from 77.75.77.39: bytes=32 time=12ms TTL=247
```

Pro reverzní vyhledání doménového názvu pomocí IP adresy slouží nástroj *nslookup*.

```
nslookup 77.75.77.39
Name:    www.seznam.cz
Address:  77.75.77.39
```

O vyhledávání DNS záznamů ve webových prohlížečích se stará operační systém, který má ve svých směrovacích záznamech uloženou IP adresu DNS serveru, který následně zajišťuje překlad.

DNS server není pouze jeden, ale jejich několik. DNS servery mohou být *veřejné* (vyhledávají záznamy uvnitř Internetu) nebo *místní* (obsahují záznamy zařízení obsažených pouze v lokální síti). Výměna informací mezi uživatelem a DNS serverem je zjednodušeně znázorněna na Obr. 4.10. DNS systém využívá již zmíněné záznamy.

- **DNS záznamy**

  A – určuje IPv4 adresu, na kterou doménový název směruje. Data v záznamu jsou pouze IP adresy ve formátu IPv4.

  AAAA – určuje IPv6 adresu, na kterou doménový název směruje. Data v záznamu jsou pouze IP adresy ve formátu IPv6.

  MX (Mail eXchange) – určuje název serveru, na který bude směrována elektronická pošta. Data v záznamu jsou textového formátu, což je doménový název serveru, kde je zprovozněn mailovací server. MX záznamů může být několik a jsou odlišovány pomocí priorit. Čím je použito nižší číslo, tím má záznam větší prioritu.

  CNAME (Canonical NAME) – umožňuje směrovat záznam z určité domény na jinou doménu. Data jsou zadávána v textovém formátu.

  TXT – slouží ke vložení libovolného textového řetězce. Využíván je převážně k ověřování vlastnictví domény.

Pro vyhledání všech zmíněných záznamů lze opět využít nástroj *nslookup*, kdy po spuštění stačí zvolit hledaný záznam.

```
nslookup
   > set q=MX      #PARAMETR HLEDANÉHO ZÁZNAMU
   > seznam.cz     #NÁZEV DOMÉNY

   Non-authoritative answer:
   seznam.cz       MX preference = 20, mail exchanger = mx2.seznam.cz
   seznam.cz       MX preference = 10, mail exchanger = mx1.seznam.cz

   mx2.seznam.cz   internet address = 77.75.76.32
   mx2.seznam.cz   AAAA IPv6 address = 2a02:598:2::32
   mx1.seznam.cz   internet address = 77.75.78.42
   mx1.seznam.cz   AAAA IPv6 address = 2a02:598:2::42
```

Získání domény a ukázka správy DNS záznamů na veřejném DNS, jsou obsaženy v příloze A.4.2.

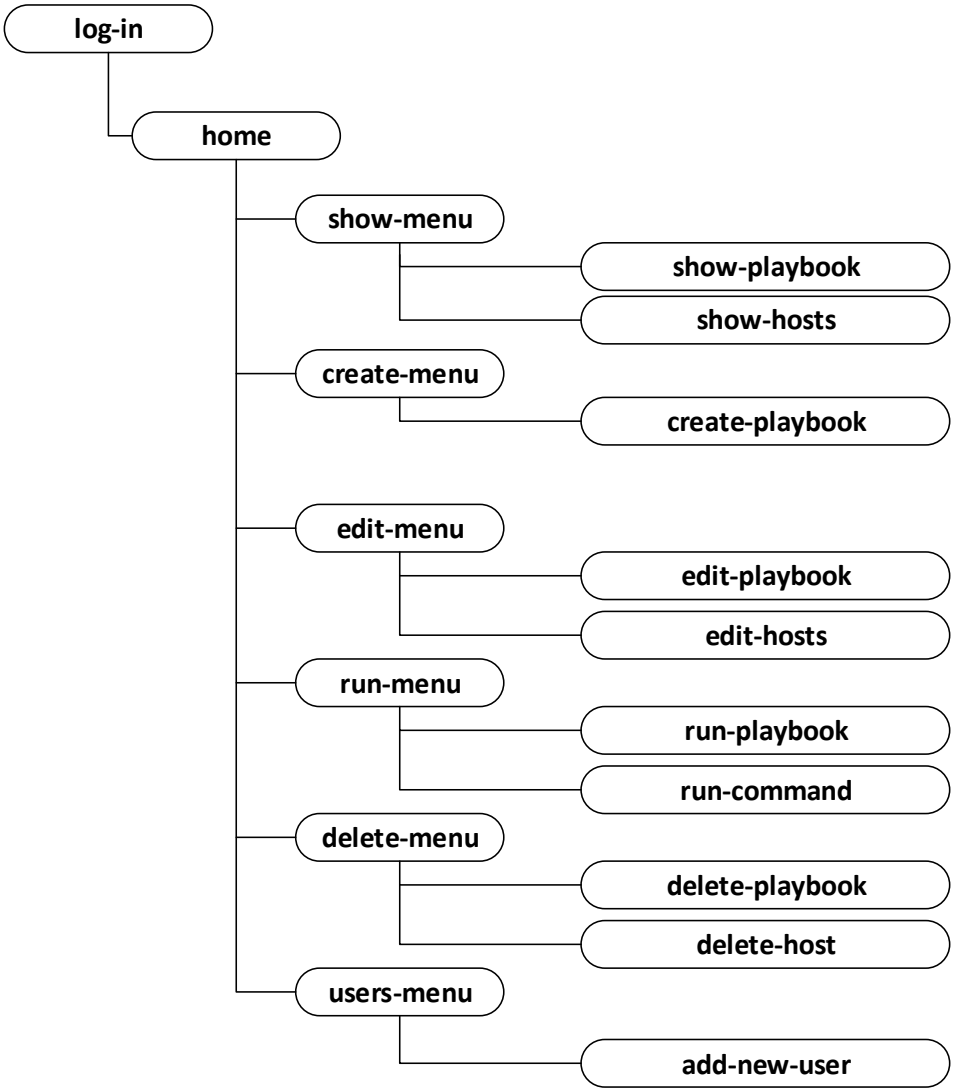
4.3 Aplikace v praxi

V této kapitole bude představena vytvořená webová aplikace umožňující vzdálenou správu počítačové sítě pomocí nástroje Ansible, který byl probrán v kapitole 2.2 a prakticky předveden v příloze A.3.

4.3.1 Ansible-web-app

Je webovou aplikací která usnadňuje práci s nástrojem Ansible, díky grafickému rozhraní. Správci počítačové sítě, dále jen uživateli, je urychlena práce při hromadné správě sítě. Uživatel nemusí vytvářet VPN (Virtual Private Network) spojení pokud se nachází mimo lokální síť. Nemusí si pamatovat přihlašovací údaje potřebné k připojení na Ansible server. Čas je ušetřen i při přechodu do adresářové struktury, kde se nachází konfigurační soubory pro obsluhu nástroje Ansible, jako jsou soubory *Playbooks* a *Hosts*. Občas se může stát, že uživatel zapomene příkazy potřebné pro spouštění playbooků nebo si nebude moci vzpomenout co je potřeba k vytvoření nového playbooku. I od těchto základních a velice důležitých informací je uživatel oproštěn.

Aplikace je rozdělena do několika částí, které obstarávají specifické funkce. Stromová struktura je vyobrazena na Obr. 4.11. Význam a funkce jednotlivých částí aplikace budou objasněny v následujícím textu. Aplikace byla navržena bez zbytečných grafických pozadí a obrázků, aby umožňovala co nejrychlejší odezvy a nebyla zpomalována načítáním výše zmíněných zdrojů.

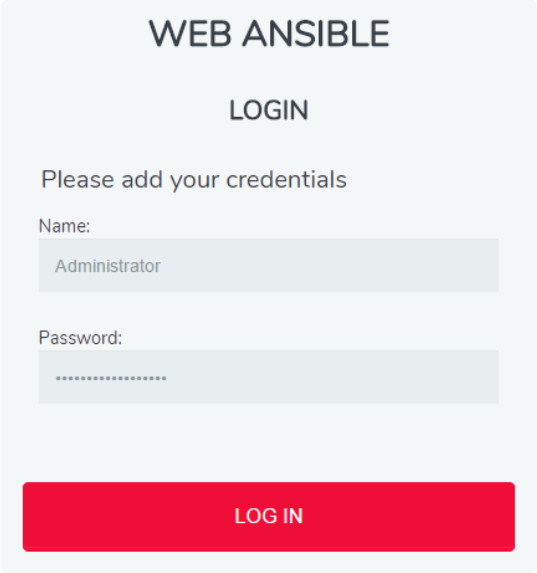


Obr. 4.11: Stromová struktura webové aplikace Ansible-web-app.

log-in

Aplikace je přístupná na adrese *vut-ansible.cz*. Uživatel je aplikací ihned přesměrován na přihlašovací stránku Obr. 4.12, kde musí uživatel zadat své přihlašovací údaje. Zadané údaje jsou nejprve hašovány pomocí jednosměrné šifrovací funkce *Bcrypt* a poté porovnány s údaji uloženými na Ansible serveru.

Pro eliminaci odposlechu hesel tzv. "mužem uprostřed" je využit protokol HTTPS, který umožňuje šifrovanou komunikaci mezi klientem a serverem. Přihlašovací údaje jsou tak na server posílány v šifrované formě pomocí šifrovacího protokolu *TLS*, jak bylo objasněno v příloze *Konfigurace HTTPS* A.4.2.



Obr. 4.12: Přihlašovací stránka.

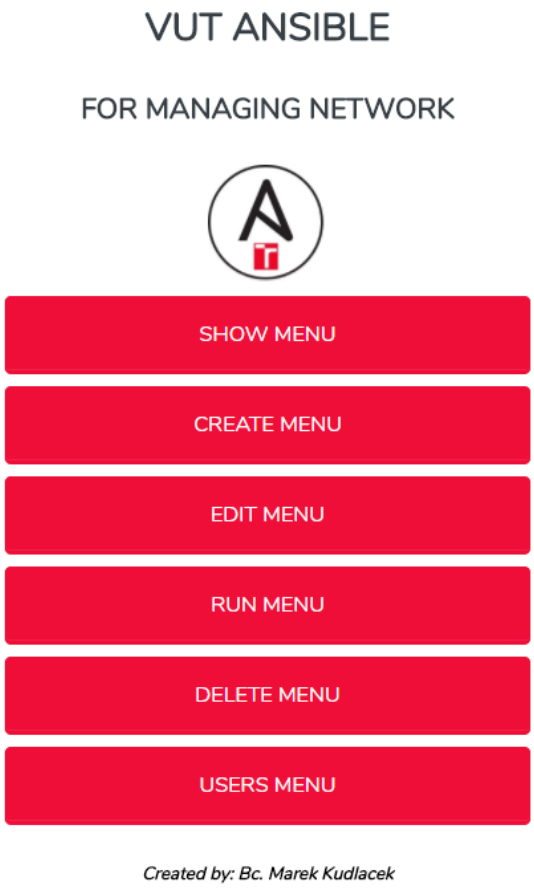
Po zdárném přihlášení je uživateli zobrazena hlavní nabídka Obr. 4.13.

home

Hlavní nabídka se nachází pod URL adresou */home*, na kterou je uživatel automaticky přesměrován pokud vlastní správné přihlašovací údaje. Tato stránka je rozcestníkem a klíčovým bodem aplikace. Uživateli nabídne ovládací prvky, kterými je možné se dostat do podsekcí umožňujících ovládání nástroje Ansible. Na Obr. 4.13 je zobrazen vzhled hlavní nabídky. Tlačítka zpřístupňují následující funkce:

- **SHOW MENU** – zobrazení seznamu playbooků a obsahu souboru hosts,
- **CREATE MENU** – vytváření nových playbooků,
- **EDIT MENU** – úprava playbooků nebo souboru hosts,

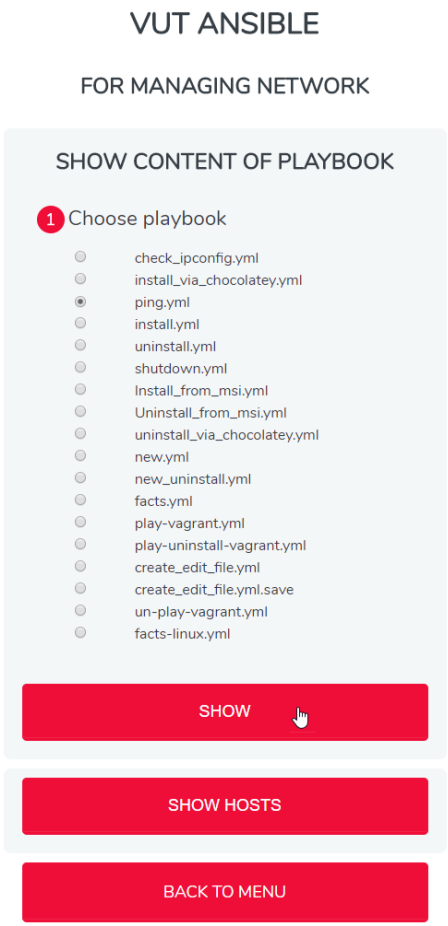
- **RUN MENU** – spouštění playbooků nebo jednotlivých modulů Ansible,
- **DELETE MENU** – mazání playbooků a hostů,
- **USERS MENU** – vytváření nových uživatelů, kteří budou moci přistupovat do webové aplikace.



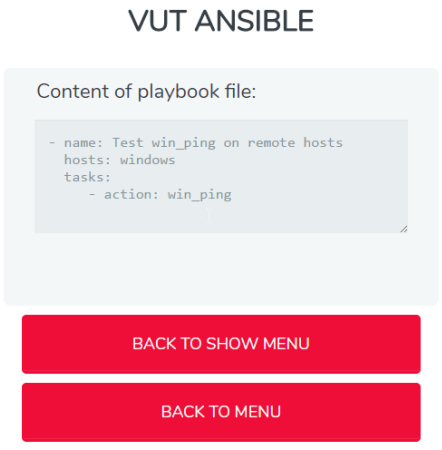
Obr. 4.13: Hlavní nabídka.

show-menu

Nabídka umožňující zobrazení obsahu vybraných playbooků nebo celkového obsahu souboru hosts. Do nabídky je možné se také dostat URL adresou */show-menu*. Uživatel je tak schopen vybrat potřebný playbook Obr. 4.14 a následně pomocí tlačítka *SHOW* zobrazit jeho obsah Obr. 4.15. Obsahy zobrazených souborů jsou pouze pro čtení a nedají se tedy nijak upravovat.



Obr. 4.14: Nabídka pro zobrazení obsahu playbooku nebo obsahu souboru hosts.

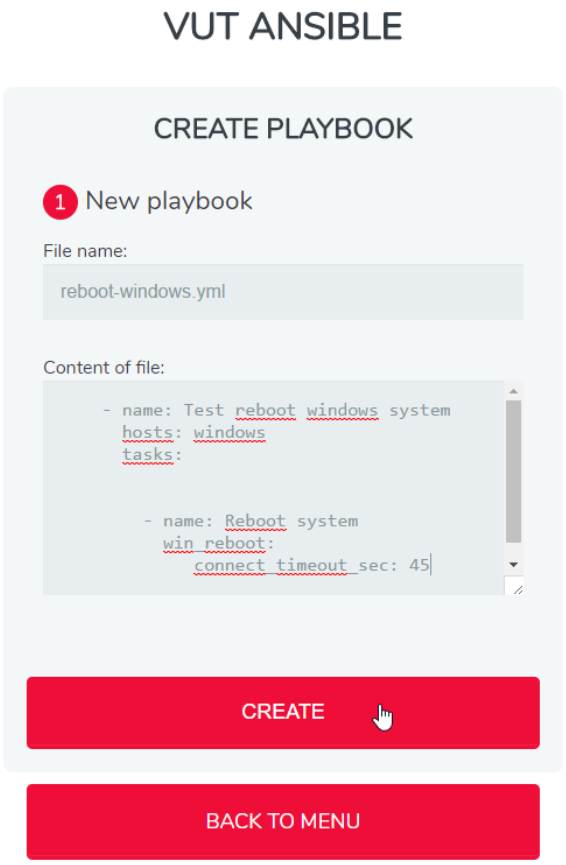


Obr. 4.15: Zobrazení obsahu playbooku ping.yml.

create-menu

Obsahuje funkci pro vytvoření nového *yaml* souboru do něhož může uživatel zapsat sled modulů a příkazů. Tato nabídka tak umožňuje vytvoření plnohodnotného playbooku Obr. 4.16.

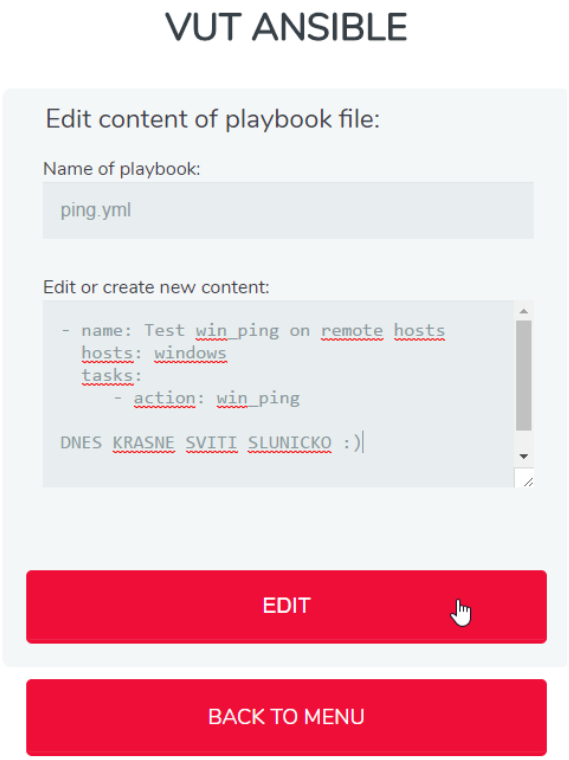
Uživatel do pole *File name* vyplní název nového playbooku a přidá mu příponu *.yml*. Do pole *Content of file* jsou vyplněny příkazy a moduly, které budou utvářet tělo playbooku. Syntaxe v těle playbooku musí být psána v *yaml* formátu. Vytvoření playbooku je uskutečněno tlačítkem *CREATE*.



Obr. 4.16: Vytvoření playbooku.

edit-menu

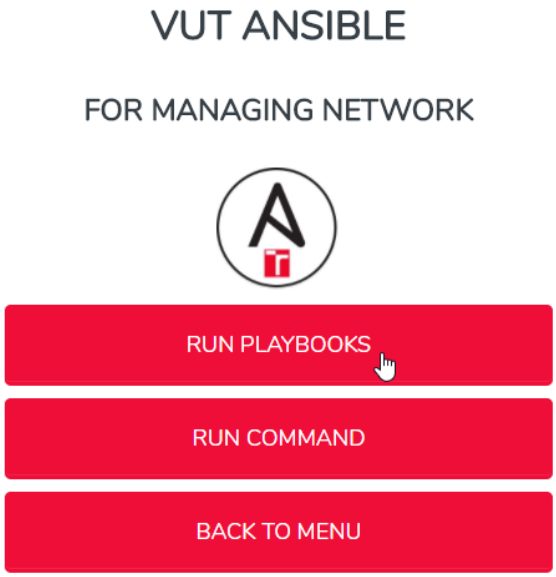
Nabídka umožňující úpravu playbooků a souboru hosts. Uživateli tak dává možnost kdykoliv upravit nebo smazat vytvořenou konfiguraci. Uživatel nejprve vybere, zda chce upravovat určitý playbook nebo zda hodlá upravovat soubor hosts. Poté je již v poli *Edit or create new content*S schopen libovolně upravovat vybraný soubor. Pro uložení upravených dat, je použito tlačítko *EDIT* jak je ukázáno na Obr. 4.17. Uživateli je poté zobrazen nově upravený obsah souboru.



Obr. 4.17: Editování playbooku ping.yml.

run-menu

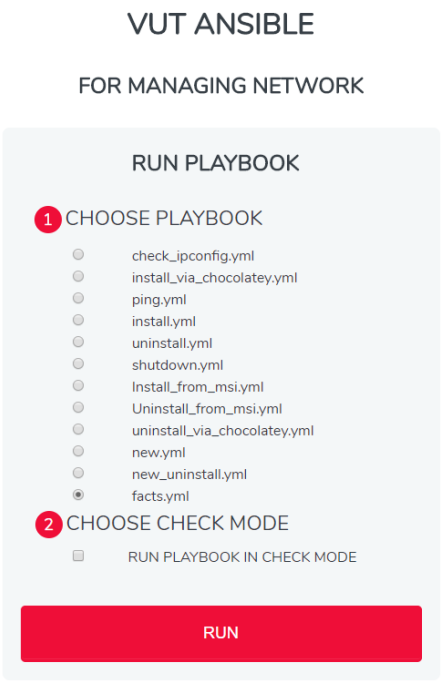
Nejdůležitější funkcí webové aplikace je spouštění playbooků a Ansible příkazů. Uživateli je nejprve nabídnuta možnost výběrů mezi spuštěním playbooku nebo samotného příkazu, jak demonstruje Obr. 4.18.



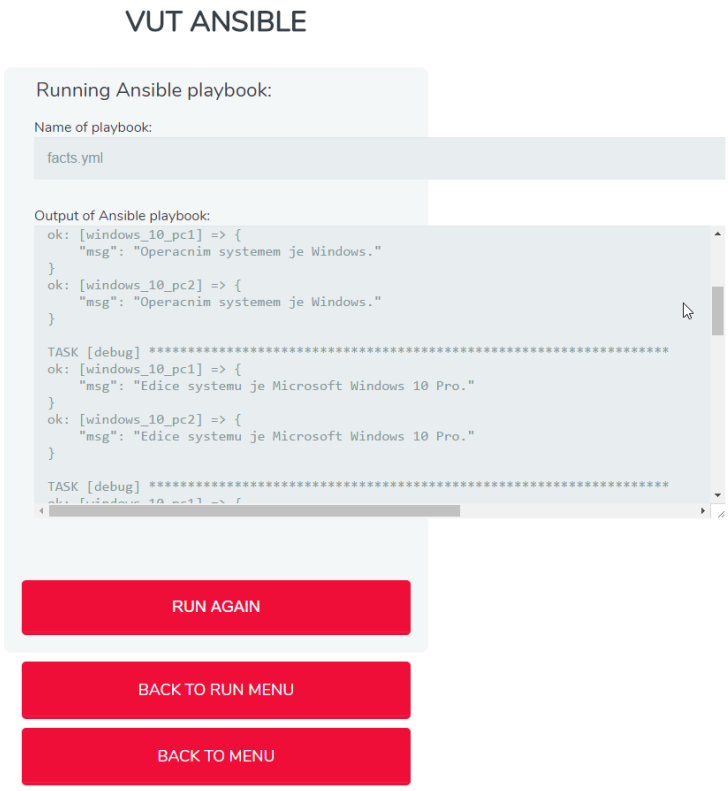
Obr. 4.18: Nabídka pro spuštění playbooku nebo samotného příkazu.

Pokud je zvoleno spuštění playbooku, je uživateli zobrazena stránka s možným výběrem playbooku a jeho následným spuštěním pomocí tlačítka *RUN*. Před spuštěním si uživatel může vybrat zda playbook nejprve nespustí v tzv. *check* módu. Playbook je pak spuštěn pouze v testovacím režimu. Na Obr. 4.19 je vyobrazeno spuštění playbooku *fact.yml*. Výstupem je stejný výpis jako při terminálovém spuštění playbooku na Ansible serveru Obr. 4.20. Spuštění vybraného playbooku je možné opakovat tlačítkem *RUN AGAIN*.

Pokud by uživatel nepotřeboval spouštět ucelený sled příkazu obsažený v playbooku, ale chtěl by spustit pouze jediný modul, může tak využít nabídky *RUN COMMAND* z Obr. 4.18. Uživatelem je poté zadán požadovaný příkaz obsahující vybraný modul a hosta, na kterého bude modul aplikován Obr. 4.21.



Obr. 4.19: Spuštění playbooku facts.yml.

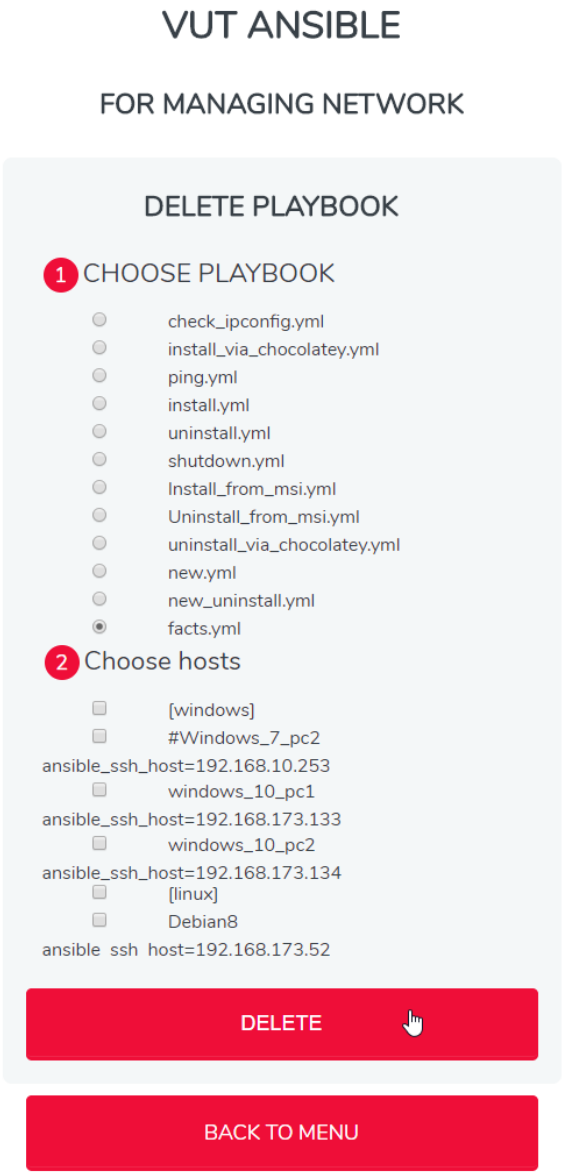


Obr. 4.20: Výstup informací ze spuštěného playbooku facts.yml.

Obr. 4.21: Spuštění modulu setup na počítač s názvem windows_10_pc1.

delete-menu

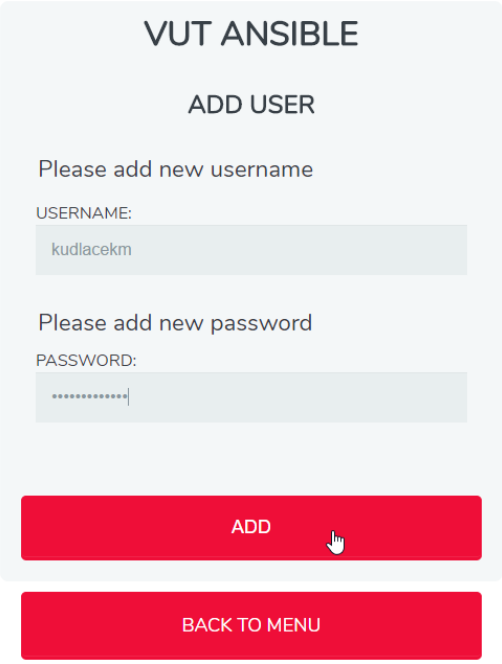
Funkce obstarávající mazání playbooků nebo jednotlivých hostů obsažených v souboru hosts. Uživatel musí označit soubor, který má být smazán. Smazání je potvrzeno tlačítkem *DELETE* jak demonstruje Obr. 4.22. Po úspěšném smazání je uživateli zobrazen buď výpis všech stávajících playbooků nebo seznam všech stávajících hostů jak je ukázáno na Obr. 4.15.



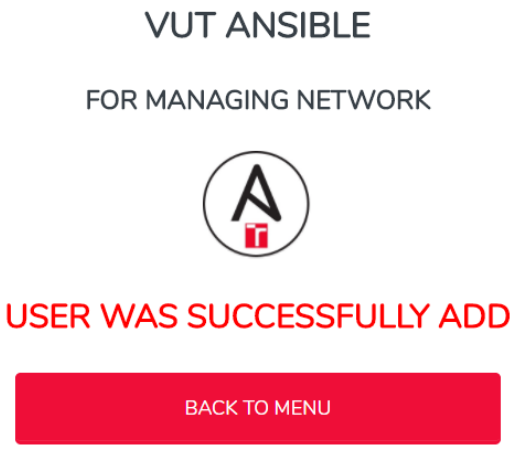
Obr. 4.22: Mazání playbooků nebo hostů.

users-menu

Poslední funkcí aplikace *Ansible-web-app* je vytváření uživatelů, kteří jsou oprávněni využívat tuto aplikaci. Nový uživatel si tak může vytvořit přihlašovací údaje obsahující jméno a heslo, viz na Obr. 4.23. Zadané heslo je zašifrováno pomocí hašovací funkce *Bcrypt*. Přihlašovací údaje jsou poté uloženy do logovacího souboru na Ansible server. Úspěšné přidání uživatele je zobrazeno na Obr. 4.24.



Obr. 4.23: Vytváření uživatelů oprávněných využívat aplikaci Ansible-web-app.



Obr. 4.24: Úspěšné přidání uživatele.

5 ZÁVĚR

V diplomové práci byla objasněna problematika systému pro správu a monitoring PC. Byly zde představeny nástroje jako Puppet, Ansible a Chef, které tuto funkcionalitu umožňují. V teoretické části byla u každého z předešlých nástrojů stručně definována funkčnost, skladba a hlavní vlastnosti.

V další části práce byl vytvořen požadovaný scénář učebny se stanicemi s různými operačními systémy, podle kterého byla následně zprovozněna testovací síť. V té se nacházely stanice využívající 64bitovou architekturu s operačním systémem Windows 10 Profesional, Windows 7 Profesional, Linuxové distribuce Ubuntu 16.04 a Debian 8. Jako systém pro správu byl vybrán nástroj Ansible.

V první polovině praktické části diplomové práce byla vytvořena testovací síť pomocí virtualizačního nástroje VMware vSphere 6.0. Hlavní virtuál server byl zprovozněn na fyzickém serveru HP Proliant DL360 G6. Ve virtuálním prostředí se nacházela převážná část stanic i s Ansible serverem. Aby testy neprobíhaly pouze na virtuálních strojích, byly použity i tři fyzické stanice (dvakrát Dell Vostro 260s a Dell Optiplex 3020), na kterých byla otestována funkčnost nástroje pro Windows 7 a 10, z důvodu převahy operačního systému Windows v reálné školní síti.

Dále je obsažen ucelený návod s postupy pro řádnou instalaci a konfiguraci nástroje Ansible, pro spolupráci s operačními systémy Windows a Linux. Dle zadání byly otestovány a popsány úkony jako instalace/odinstalace aplikací, zjištění informací o stroji, vzdálený restart a vypnutí, tvorba uživatelů se specifikací oprávnění a další, které byly rozčleněny do playbooků. Bylo ověřeno, že vybraný nástroj je vhodný pro praktickou implementaci do reálné sítě.

Ve druhé polovině praktické části byl pro nástroj Ansible vytvořen nadstavbový systém umožňující správu z lokální nebo veřejné sítě. Tímto nástrojem je webová aplikace Ansible-web-app, která je napsána v programovacím jazyce Java s využitím frameworku Spring.

Aby mohla být aplikace dostupná z lokální i veřejné sítě, byl na Ansible serveru zprovozněn webový server Apache. Dále byly provedeny potřebné zásahy do konfigurace síťového firewallu a firewallu na Ansible serveru. Posledním bodem bylo zavedení záznamu o doméně www.vut-ansible.cz ve veřejném DNS, pod kterým je webová aplikace dostupná. Ke všem zmíněným bodům byla vytvořena podrobná dokumentace seznamující s problematikou a s kroky, které jsou pro provoz webové aplikace nezbytné.

Nakonec byla aplikace zdárně otestována v praxi, kdy na virtuální a fyzické stanice byly použity vytvořené playbooky z teoretické části. Byla tak ověřena funkčnost nástroje Ansible ovládaného přes webovou aplikaci Ansible-web-app jak z lokální, tak i z veřejné sítě.

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

SSH	Protokol umožňující šifrovanou vzálenou správu zařízení – Secure Shell
WinRM	Služba umožňující šifrovanou i nešifrovanou vzdálenou správu zařízení s odlišným HW a SW. Ke své komunikaci využívá WS-Management protokol – Windows Remote Management
FQDN	Služba umožňující specifikovat plně doménové jméno počítače nebo serveru – Fully Qualified Domain Name
CM	Označení nástrojů, umožňujících automatizaci správy sítě – Configuration Management
Ruby	Objektově orientovaný skriptovací jazyk, konkurent jazyka Python a Perl
Push	Metoda, kdy server zasílá informace podřízeným strojům, aniž by si je vyžádali
YAML	Formát pro serializaci strukturovaných dat – Ain't Markup Language
Python	Univerzální programovací jazyk
Rpm	Souborový formát, který využívá např. CentOS a Fedora – Red Hat Package Manager
Deb	Souborový formát, který využívá např. Debian a Ubuntu
Yum	Nástroj pro práci se soubory formátu .rpm, využívá např. CentOS – The Yellowdog Updated, Modified
Dnf	Nástroj pro práci se soubory formátu .rpm. Nástupce yum, využívá např. Fedora – Dandified Yum
Apt	Nástroj pro práci se soubory formátu .deb, využívá např. Debian a Ubuntu – Advanced Package Tool
HTTPS	Zabezpečená forma protokolu HTTP pro komunikaci mezi serverem a klientskou částí – Hypertext Transfer Protocol Secure
VPN	Virtuální síť umožňující propojení dvou lokálních sítí napříč veřejnou sítí (Internet) – Virtual Private Network
GUI	Uživatelské rozhraní zajišťující obsluhu systému pomocí grafických ovládacích prvků – Graphical User Interface

URL	Řetězec znaků určený pro specifikaci zdrojů informací umístěných na Internetu – Uniform Resource Locator
DNS	Systém pro překlad IP adres na doménové názvy. Například IP *77.75.77.39* odpovídá doménovému názvu *www.seznam.cz* – Domain Name System
PHP	Programovací jazyk určený převážně na tvorbu dynamických webových stránek – Hypertext Preprocessor
JAVA	Celosvětově nejrozšířenější objektově orientovaný programovací jazyk, který je využit od počítačového, až po automobilový průmysl
JSP	Technologie pro vývoj dynamických HTML stránek v jazyce Java – Java Server Pages
IPv4	Technologie využívána pro směrování komunikační sítí, za využití 32 bitového čísla zapsaného v dekadickém formátu, např. 192.168.10.11 – Internet Protocol version 4
IPv6	Obdoba IPv4 , využívá 128 bitového čísla zapsaného v šestnáctkovém formátu, např. ce01:2c01:0000:0000:65a0:123c:40aa:0101 – Internet Protocol version 6
Bcrypt	Šifrovací funkce pro jednosměrné šifrování hesel, postavena na základech symetrické blokové šifry Blowfish. V dnešní době je brána za jednu z nejbezpečnějších šifrovacíh funkcí.

SEZNAM PŘÍLOH

A PŘÍLOHY TESTOVACÍ SÍŤ

Tato část textu obsahuje přílohy, které spadají do kapitoly 3.

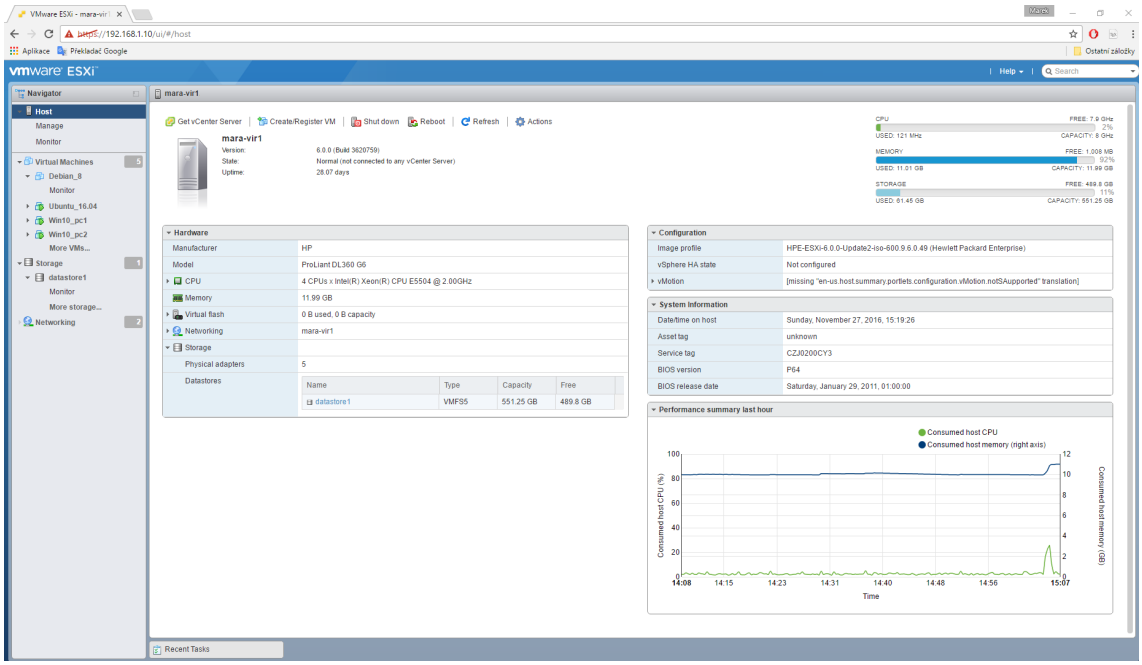
A.1 SW stroje (VMware vSphere 6.0) a HW

Grafická dokumentace ve které jsou obsaženy informace o virtualizačním nástroji VMware v Sphere 6.0.

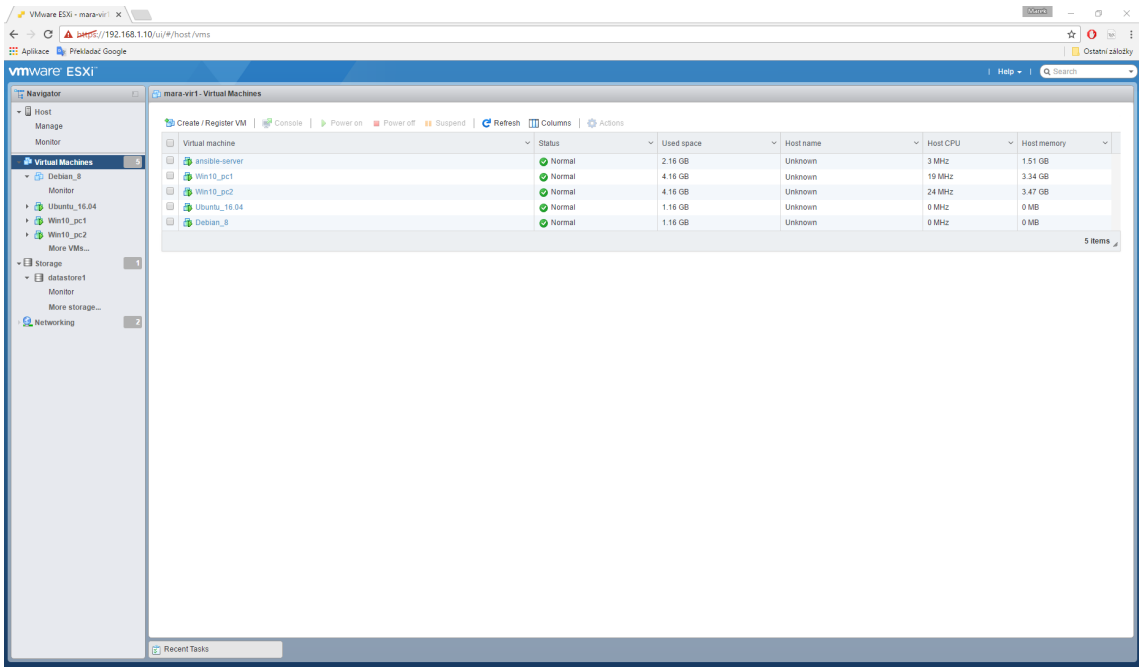
(a) Fyzický server.

(b) Detail serveru HP Proliant DL360 G6.

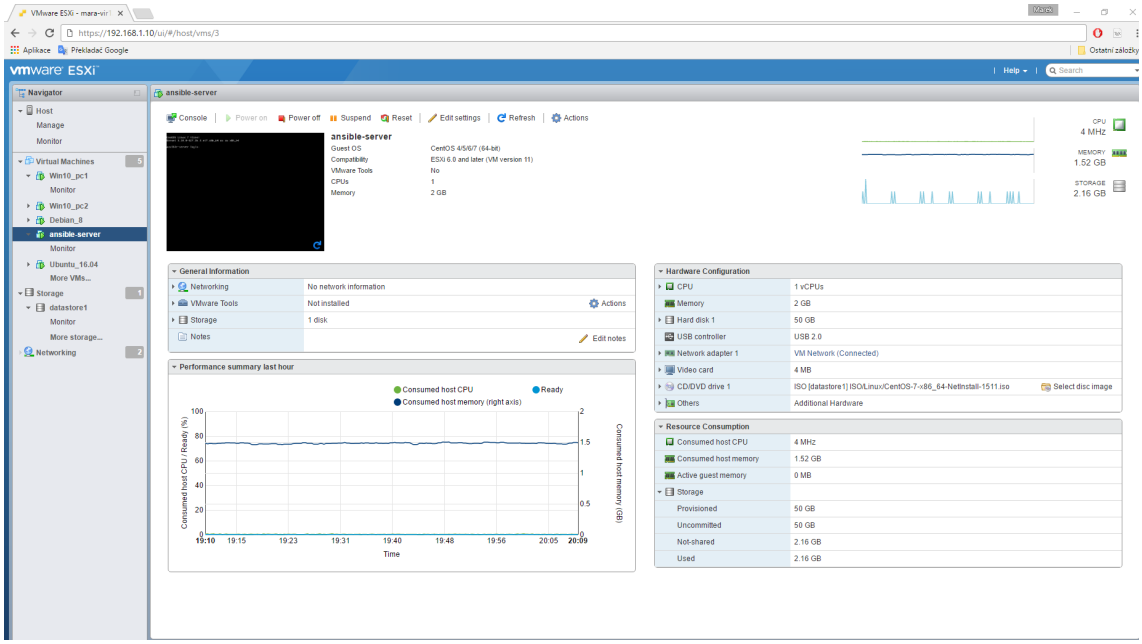
Obr. A.1: Umístění serveru HP Proliant DL360 G6 v serverovně.



Obr. A.2: Webové rozhraní VMware vSphere 6.0.



Obr. A.3: Webové rozhraní VMware vSphere 6.0 a vytvořené virtuální stanice.



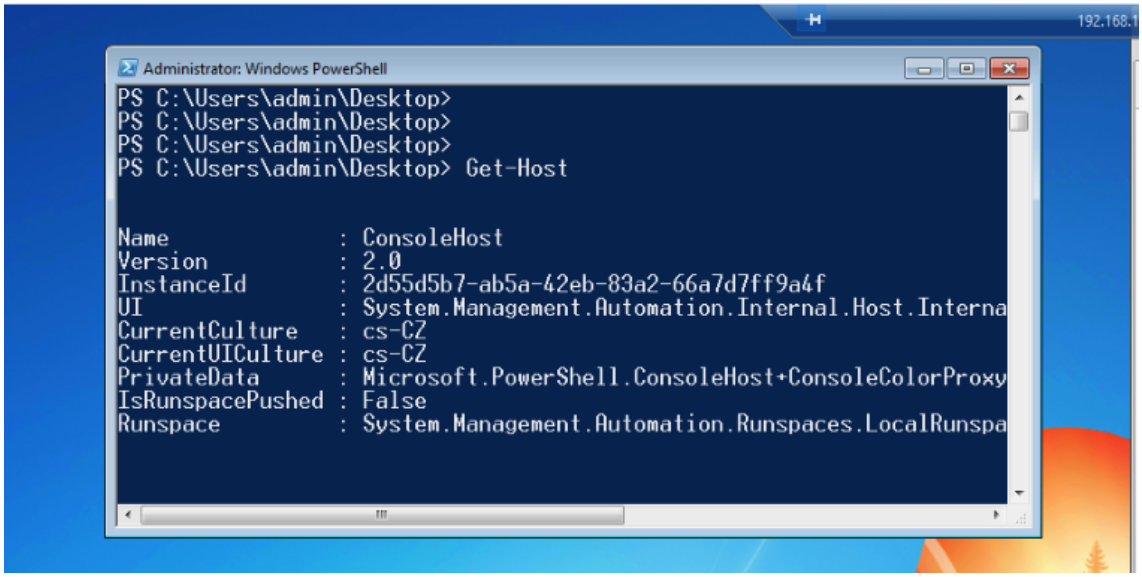
Obr. A.4: Ansible server nainstalovaný ve VMware vSphere 6.0.

Obr. A.5: Stolní počítače pro Windows 7 a Windows 10.

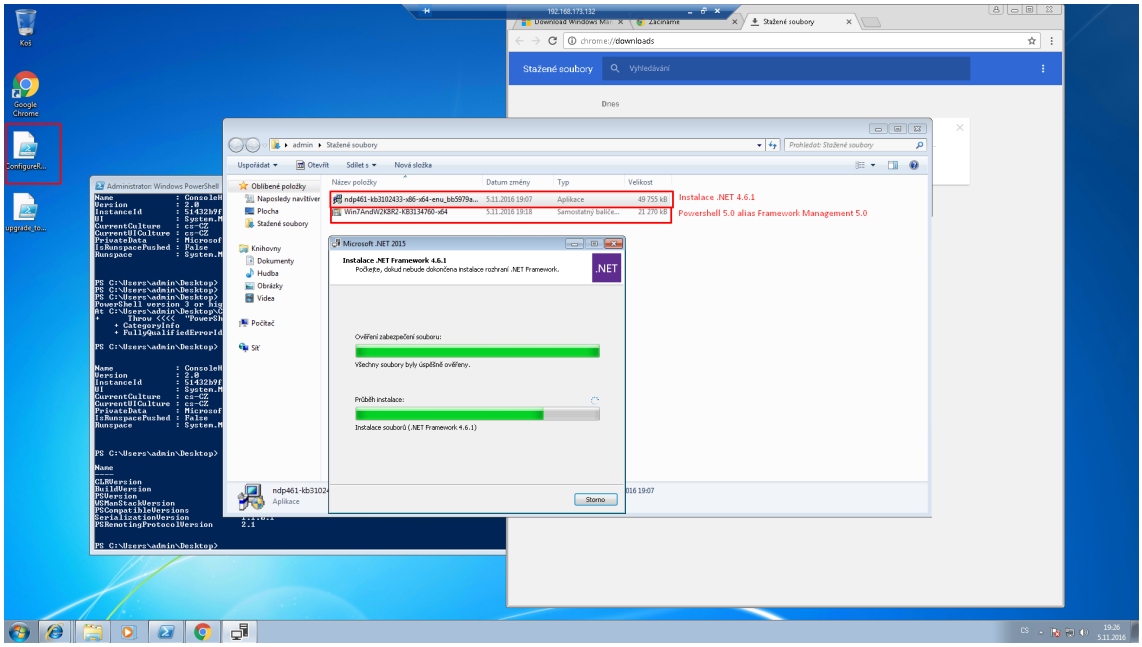
A.2 Ansible v testovací síti

Dokumentace obsahující informace potřebné k instalaci, nastavení a provozu Ansible v testovací síti.

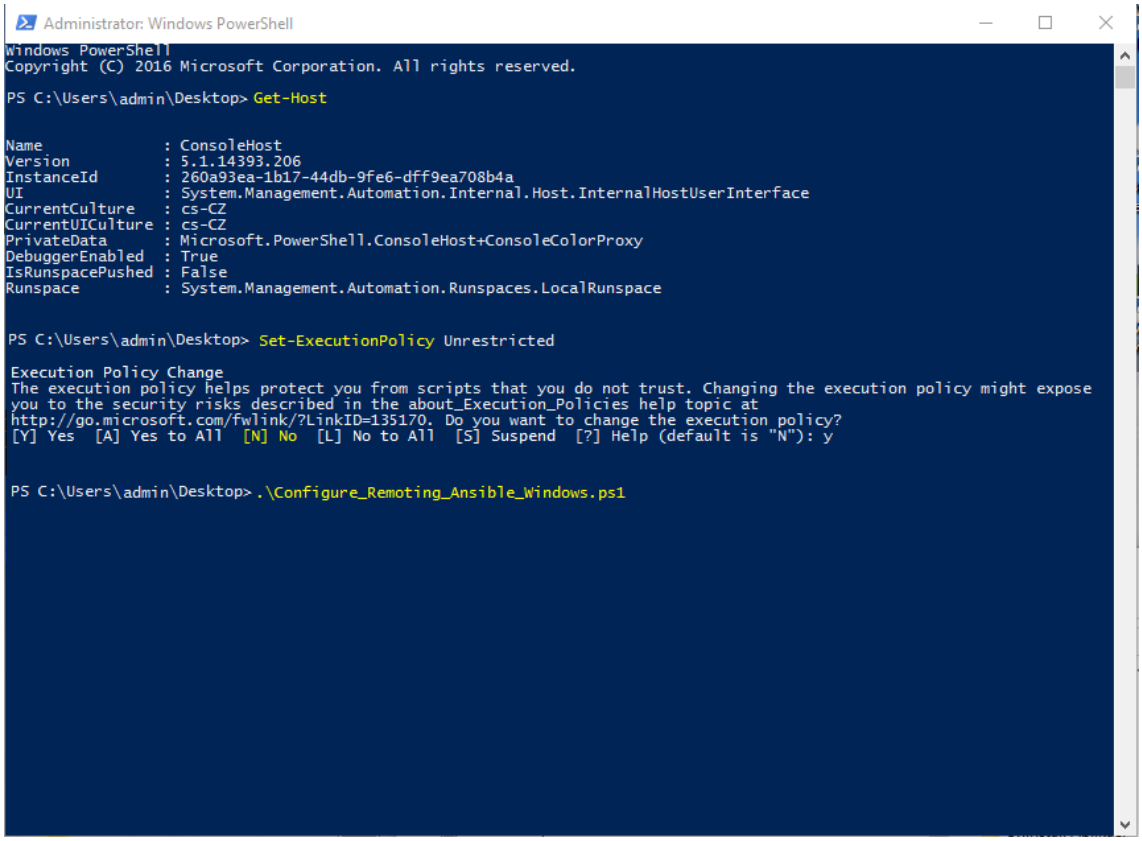
A.2.1 Ansible na Windows



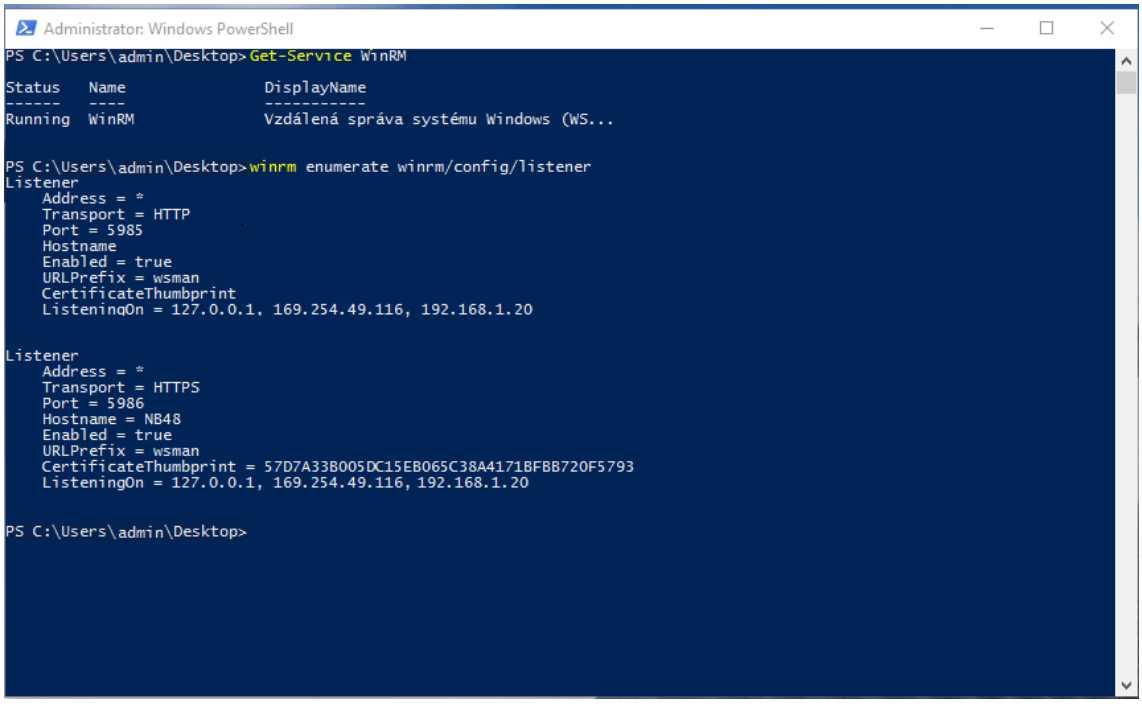
Obr. A.6: Zjištění verze Powershell na stanici Windows_7_pc1.



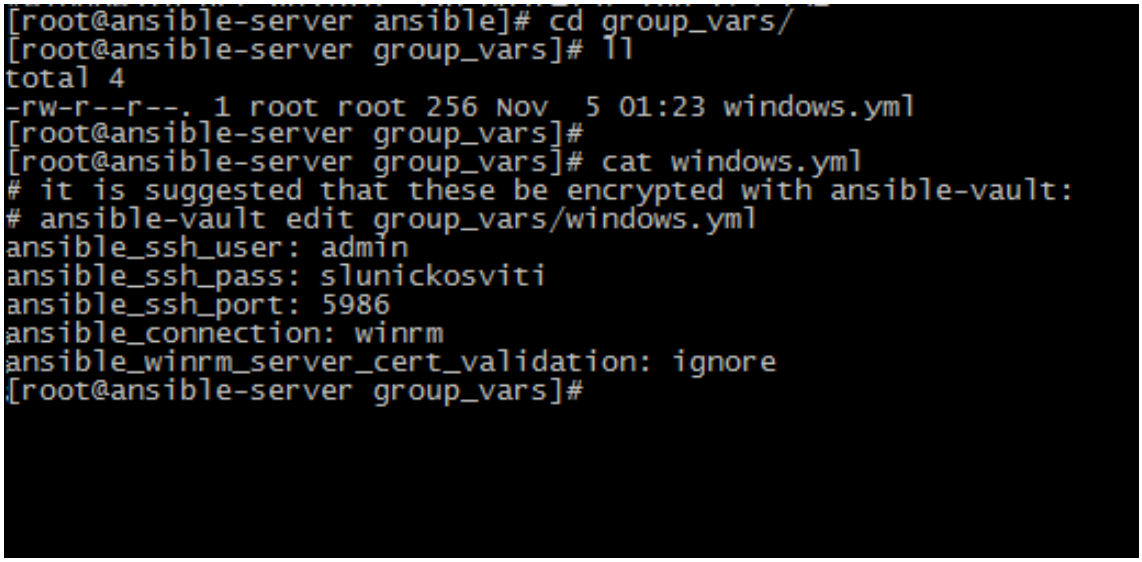
Obr. A.7: Instalace .NET 4.6.1 a Powershell 5.0.



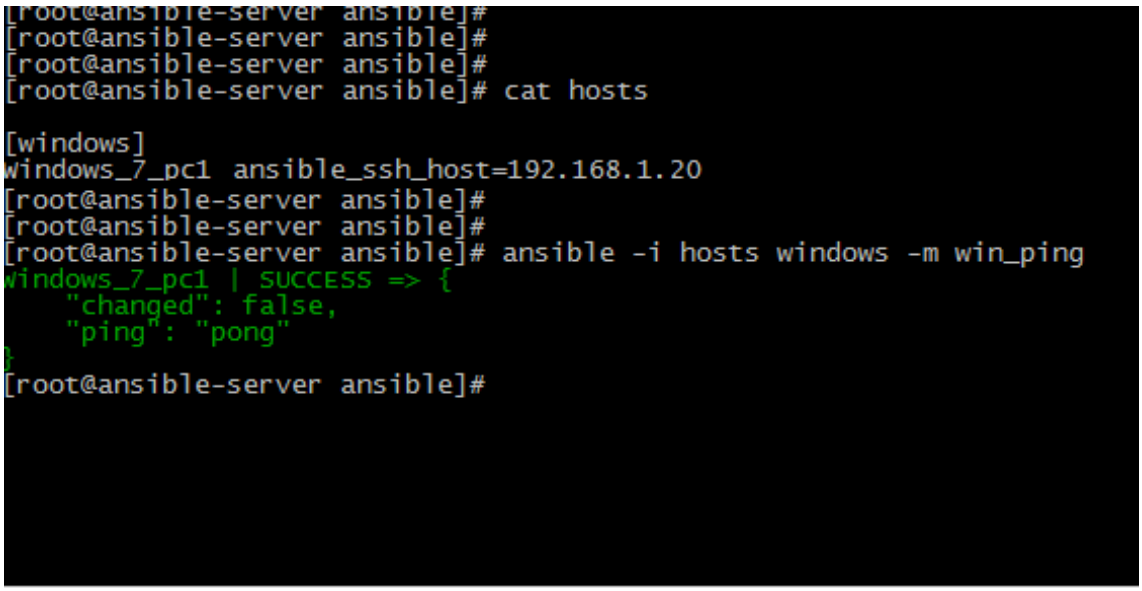
Obr. A.8: Ověření verze PS 5.0 a spuštění konfiguračního skriptu pro Ansible.



Obr. A.9: Ověření spuštění služby WinRM.



Obr. A.10: Konfigurační soubor windows.yml, potřebný k navázání komunikace.



Obr. A.11: Ověření komunikace se stanicí Windows_7_pc1, modul win_ping.

Powershell skript *Configure_Remoting_Ansible_Windows.ps1*, pro automatické vytvoření certifikátu, povolení a nastavení služby WinRM.

```
# Configure a Windows host for remote management with Ansible
# -----------------------------------------------------------
#
# This script checks the current WinRM/PSRemoting configuration and makes the
# necessary changes to allow Ansible to connect, authenticate and execute
# PowerShell commands.
#
# Set $VerbosePreference = "Continue" before running the script in order to
# see the output messages.
# Set $SkipNetworkProfileCheck to skip the network profile check.  Without
# specifying this the script will only run if the device's interfaces are in
# DOMAIN or PRIVATE zones.  Provide this switch if you want to enable winrm on
# a device with an interface in PUBLIC zone.
#
# Set $ForceNewSSLCert if the system has been syspreped and a new SSL Cert
# must be forced on the WinRM Listener when re-running this script. This
# is necessary when a new SID and CN name is created.
#
# Written by Trond Hindenes <trond@hindenes.com>
# Updated by Chris Church <cchurch@ansible.com>
# Updated by Michael Crilly <mike@autologic.cm>
# Updated by Anton Ouzounov <Anton.Ouzounov@careerbuilder.com>
#
# Version 1.0 - July 6th, 2014
# Version 1.1 - November 11th, 2014
# Version 1.2 - May 15th, 2015
# Version 1.3 - April 4th, 2016

Param (
    [string]$SubjectName = $env:COMPUTERNAME,
    [int]$CertValidityDays = 365,
    [switch]$SkipNetworkProfileCheck,
    $CreateSelfSignedCert = $true,
    [switch]$ForceNewSSLCert
)

Function New-LegacySelfSignedCert
{
    Param (
        [string]$SubjectName,
        [int]$ValidDays = 365
    )

    $name = New-Object -COM "X509Enrollment.CX500DistinguishedName.1"
    $name.Encode("CN=$SubjectName", 0)

    $key = New-Object -COM "X509Enrollment.CX509PrivateKey.1"
    $key.ProviderName = "Microsoft RSA SChannel Cryptographic Provider"
    $key.KeySpec = 1
    $key.Length = 1024
    $key.SecurityDescriptor = "D:PAI(A;;0xd01f01ff;;;SY)(A;;0xd01f01ff;;;BA)(A;;0x80120089;;;NS)"
    $key.MachineContext = 1
    $key.Create()

    $serverauthoid = New-Object -COM "X509Enrollment.CObjectId.1"
```

```
    $serverauthoid.InitializeFromValue("1.3.6.1.5.5.7.3.1")
    $ekuoids = New-Object -COM "X509Enrollment.CObjectIds.1"
    $ekuoids.Add($serverauthoid)
    $ekuext = New-Object -COM "X509Enrollment.CX509ExtensionEnhancedKeyUsage.1"
    $ekuext.InitializeEncode($ekuoids)

    $cert = New-Object -COM "X509Enrollment.CX509CertificateRequestCertificate.1"
    $cert.InitializeFromPrivateKey(2, $key, "")
    $cert.Subject = $name
    $cert.Issuer = $cert.Subject
    $cert.NotBefore = (Get-Date).AddDays(-1)
    $cert.NotAfter = $cert.NotBefore.AddDays($ValidDays)
    $cert.X509Extensions.Add($ekuext)
    $cert.Encode()

    $enrollment = New-Object -COM "X509Enrollment.CX509Enrollment.1"
    $enrollment.InitializeFromRequest($cert)
    $certdata = $enrollment.CreateRequest(0)
    $enrollment.InstallResponse(2, $certdata, 0, "")

    # Return the thumbprint of the last installed certificate;
    # This is needed for the new HTTPS WinRM listerner we're
    # going to create further down.
    Get-ChildItem "Cert:\LocalMachine\my"| Sort-Object NotBefore -Descending | Select -First 1
                | Select -Expand Thumbprint
}

# Setup error handling.
Trap
{
    $_
    Exit 1
}
$ErrorActionPreference = "Stop"

# Detect PowerShell version.
If ($PSVersionTable.PSVersion.Major -lt 3)
{
    Throw "PowerShell version 3 or higher is required."
}

# Find and start the WinRM service.
Write-Verbose "Verifying WinRM service."
If (!(Get-Service "WinRM"))
{
    Throw "Unable to find the WinRM service."
}
ElseIf ((Get-Service "WinRM").Status -ne "Running")
{
    Write-Verbose "Starting WinRM service."
    Start-Service -Name "WinRM" -ErrorAction Stop
    Write-Verbose "Setting WinRM service to start automatically on boot."
    Set-Service -Name "WinRM" -StartupType Automatic
}

# WinRM should be running; check that we have a PS session config.
If (!(Get-PSSessionConfiguration -Verbose:$false) -or
(!(Get-ChildItem WSMan:\localhost\Listener)))
{
```

```
  if ($SkipNetworkProfileCheck) {
    Write-Verbose "Enabling PS Remoting without checking Network profile."
    Enable-PSRemoting -SkipNetworkProfileCheck -Force -ErrorAction Stop
  }
  else {
    Write-Verbose "Enabling PS Remoting"
    Enable-PSRemoting -Force -ErrorAction Stop
  }
}
Else
{
    Write-Verbose "PS Remoting is already enabled."
}

# Make sure there is a SSL listener.
$listeners = Get-ChildItem WSMan:\localhost\Listener
If (!($listeners | Where {$_.Keys -like "TRANSPORT=HTTPS"}))
{
    # HTTPS-based endpoint does not exist.
    If (Get-Command "New-SelfSignedCertificate" -ErrorAction SilentlyContinue)
    {
        $cert = New-SelfSignedCertificate -DnsName $SubjectName -CertStoreLocation
                                "Cert:\LocalMachine\My"
        $thumbprint = $cert.Thumbprint
        Write-Host "Self-signed SSL certificate generated; thumbprint: $thumbprint"
    }
    Else
    {
        $thumbprint = New-LegacySelfSignedCert -SubjectName $SubjectName
        Write-Host "(Legacy) Self-signed SSL certificate generated; thumbprint: $thumbprint"
    }

    # Create the hashtables of settings to be used.
    $valueset = @{}
    $valueset.Add('Hostname', $SubjectName)
    $valueset.Add('CertificateThumbprint', $thumbprint)

    $selectorset = @{}
    $selectorset.Add('Transport', 'HTTPS')
    $selectorset.Add('Address', '*')

    Write-Verbose "Enabling SSL listener."
    New-WSManInstance -ResourceURI 'winrm/config/Listener' -SelectorSet
                $selectorset -ValueSet $valueset
}
Else
{
    Write-Verbose "SSL listener is already active."

    # Force a new SSL cert on Listener if the $ForceNewSSLCert
    if($ForceNewSSLCert){

        # Create the new cert.
        If (Get-Command "New-SelfSignedCertificate" -ErrorAction SilentlyContinue)
        {
            $cert = New-SelfSignedCertificate -DnsName $SubjectName -CertStoreLocation
                                              "Cert:\LocalMachine\My"
            $thumbprint = $cert.Thumbprint
            Write-Host "Self-signed SSL certificate generated; thumbprint: $thumbprint"
```

```
        }
        Else
        {
            $thumbprint = New-LegacySelfSignedCert -SubjectName $SubjectName
            Write-Host "(Legacy) Self-signed SSL certificate generated; thumbprint: $thumbprint"
        }

        $valueset = @{}
        $valueset.Add('Hostname', $SubjectName)
        $valueset.Add('CertificateThumbprint', $thumbprint)

        # Delete the listener for SSL
        $selectorset = @{}
        $selectorset.Add('Transport', 'HTTPS')
        $selectorset.Add('Address', '*')
        Remove-WSManInstance -ResourceURI 'winrm/config/Listener' -SelectorSet $selectorset

        # Add new Listener with new SSL cert
        New-WSManInstance -ResourceURI 'winrm/config/Listener' -SelectorSet $selectorset
                                -ValueSet $valueset
    }
}

# Check for basic authentication.
$basicAuthSetting = Get-ChildItem WSMan:\localhost\Service\Auth | Where {$_.Name -eq "Basic"}
If (($basicAuthSetting.Value) -eq $false)
{
    Write-Verbose "Enabling basic auth support."
    Set-Item -Path "WSMan:\localhost\Service\Auth\Basic" -Value $true
}
Else
{
    Write-Verbose "Basic auth is already enabled."
}

# Configure firewall to allow WinRM HTTPS connections.
$fwtest1 = netsh advfirewall firewall show rule name="Allow WinRM HTTPS"
$fwtest2 = netsh advfirewall firewall show rule name="Allow WinRM HTTPS" profile=any
If ($fwtest1.count -lt 5)
{
    Write-Verbose "Adding firewall rule to allow WinRM HTTPS."
    netsh advfirewall firewall add rule profile=any name="Allow WinRM HTTPS"
                dir=in localport=5986 protocol=TCP action=allow
}
ElseIf (($fwtest1.count -ge 5) -and ($fwtest2.count -lt 5))
{
    Write-Verbose "Updating firewall rule to allow WinRM HTTPS for any profile."
    netsh advfirewall firewall set rule name="Allow WinRM HTTPS" new profile=any
}
Else
{
    Write-Verbose "Firewall rule already exists to allow WinRM HTTPS."
}

# Test a remoting connection to localhost, which should work.
$httpResult = Invoke-Command -ComputerName "localhost" -ScriptBlock {$env:COMPUTERNAME}
-ErrorVariable httpError -ErrorAction SilentlyContinue
$httpsOptions = New-PSSessionOption -SkipCACheck -SkipCNCheck -SkipRevocationCheck
```

```
$httpsResult = New-PSSession -UseSSL -ComputerName "localhost" -SessionOption $httpsOptions
-ErrorVariable httpsError -ErrorAction SilentlyContinue

If ($httpResult -and $httpsResult)
{
    Write-Verbose "HTTP: Enabled | HTTPS: Enabled"
}
ElseIf ($httpsResult -and !$httpResult)
{
    Write-Verbose "HTTP: Disabled | HTTPS: Enabled"
}
ElseIf ($httpResult -and !$httpsResult)
{
    Write-Verbose "HTTP: Enabled | HTTPS: Disabled"
}
Else
{
    Throw "Unable to establish an HTTP or HTTPS remoting session."
}
Write-Verbose "PS Remoting has been successfully configured for Ansible."
```

A.2.2 Ansible na Linux

Soukromý klíč Ansible serveru.

```
     -----BEGIN RSA PRIVATE KEY-----
MIIEowIBAAKCAQEAobFroBfVNed1yM5qXdAiZHdm94F/e38HvzwKLop+bBTuqfZO
zT9VsoY64VcL14hrpcLODHV6GYfcBNs2SpJ2nqaY5SvoMuVRgYiYzDHRjeXQTpma
R5DOMdpmVZTQZIn97jn2OKRPDAe3+fW1zFAO48TESqTk9DYYRLeXupqGtbF+w7hi
Z3RZOOC4Wan/Afr0F+3HYEKcE42B6oaYAs7XPZGB/xNCJ8TghGJm3YFMd/eZnodp
PzB1BiBc1zclfB9l6UYWmuwz5TEeeBUqhs+izTU5u+5Gx8uKs1jh+Rh49Kv2qp57
qbP3MfLrarFwUstkZq6hJxLTWNsK9BuayhZUkwIDAQABAoIBAAWYej0EFiDTbcQR
QKUWeto3N50BIkRH1SDNFx0dNdrPrqK7H2cC9hhDhO4fcQ14byFBaJgNyLfxd6/D
VlGmV+yNn2Tu7A422Wc+LuuCHdpnRZoyvqATaCABiuETUVLpY5xG7T+zL/kem0NP
OYdkjd9cA98Bbdr1tElMgTe3+Ul4WPFsbYPRDR2jHDTso+mDvQO/lOTVCHqCXaMe
zipenABCTosv9LLFxwVQTZjrNVog3F5JZK48SCH7Q6cq5WtTT8cb53+WWyYFJnLc
KJcmPhz7vcDptsntNtqXXArJ1vfyFTAhddiqaUAmXhXtW+RpZErFv0ORzrqhPUX0
+jeIdzwCcxiiNkA8Hg7NJUCpyHj1Kh1ITnc+H7jfWCfCkqw1U2CIunB2yNcZMQ7t
3LLuGnTDzKGhpUOPTzkDYk149zSK2YIgN8FUbRNgvtfoWnffEsQPHE+ZXCZIqbcp
rP5Q/vfqU7bbAW/IlXwwPf5h/S/2uj3TRCu+7+cCgYEAwQ7/WavIBcnFt/b4q5N9
RoHxH4Vq6fSqszVP31VD/nSgH6j0eoUBSWgEHLU5vWSbv2624Xsh514qWoPKN2fg
A26AzkB2THjot41RWfKyM5wQAbsvRbrWYp1xSt33uMT7LjKF44rvvV50haJwSKhA
Yfze+k0N12gb9+tsb++vxAOCgYAqNNOOWu+JXAaL3DdpP3MlQymj1wieD8pH1Ca7
0dov47qdGKeUHOGLynaW90YUESDML36Y9syapcn0BF4xnJZGrA/pIa/6FYPKINLT
xwYRdozXNiM4u5MfqDohUyBGwApyHwJVzpqMNOmEN6WxlKfTwG2I7DIBhI/YNV50
LNsYKwKBgD15v3Zv0xcri/d7zgQZG1WFi3gHyCHtNynREw0DwFe8QNUTj78atsYu
EjQ8pEDit73i069gEKQIs2MP/fvElqZSSkRDkMdV9ikSTYfg4RwkBPjLsBnEwXR+
rV11FnO17eEPLnsB4OgF1mMe5u2l9SQBiEtY3zUMH6zKOz+S5KMJ
-----END RSA PRIVATE KEY-----
```

Veřejný klíč Ansible serveru.

```
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQChsWugF9U153XIzmpdOCJkd2b3gX
97fwe/PAouin5sFO6p9k7NP1WyhjrhVwvXiGulws4MdXoZh9wE2zZKknaeppjlK+gy5
VGBiJjMMdGN5dBOmZpHkPQx2mZVlNBkif3uOfbQpE8MB7f59bXMUA7jxMRKpOTONhhE
t5e6moa1sX7DuGJndFk44LhZqf8B+vQX7cdgQpwTjYHqhpgCztc9kYH/EOInxOCEYmb
dgUx395meh2k/MHUGIFzXNyV8H2XpRhaa7DPlMR54FSqGz6LNNTm77kbHy4qzWOH5GHj
Oq/aqnnups/cx8utqsXBSy2RmrqEnEtNY2wrOG5rKFlST root@ansible-server
```

```
[root@ansible-server ~]# ssh-copy-id root@Debian8
The authenticity of host 'root@Debian8 (192.168.1.50)' can't be established.
RSA key fingerprint is 6a:73:03:63:2e:64:0a:19:c3:bb:40:7a:90:79:fa:72.
Are you sure you want to continue connecting (yes/no)? yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys
root@Debian8 password:
Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'root@Debian_8'"
and check to make sure that only the key(s) you wanted were added.

[root@ansible-server ~]# ssh root@Debian_8
root@Debian8:~#
```

Obr. A.12: Inicializace SSH spojení Ansible serveru s Debian_8.

```
[root@ansible-server ansible]# ansible -i hosts linux -m ping
Debian8| SUCCESS => {
    cnanged": false,
    "ping": "pong"
}
[root@ansible-server ansible]#
```

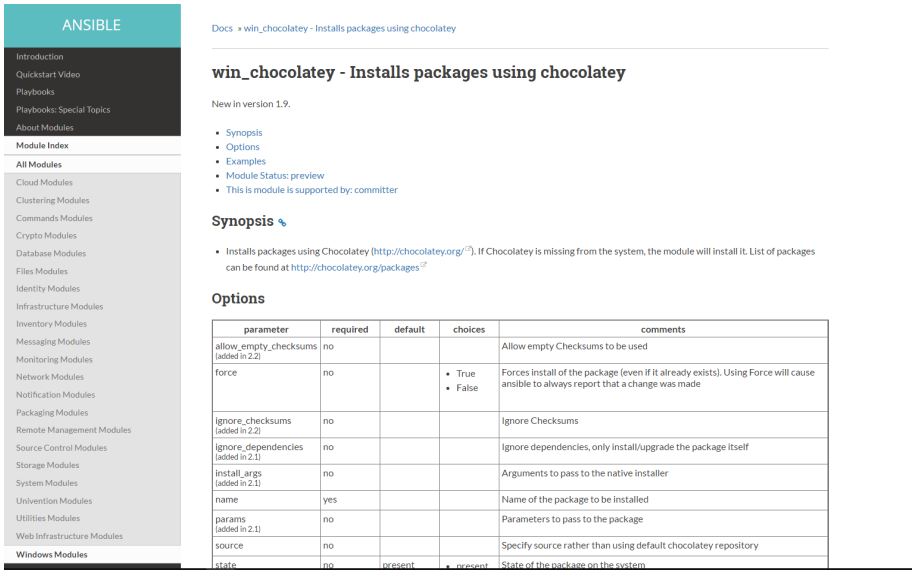
Obr. A.13: Ověření komunikace se stanicí Debian_8, modul ping.

A.3 Implementace zadání

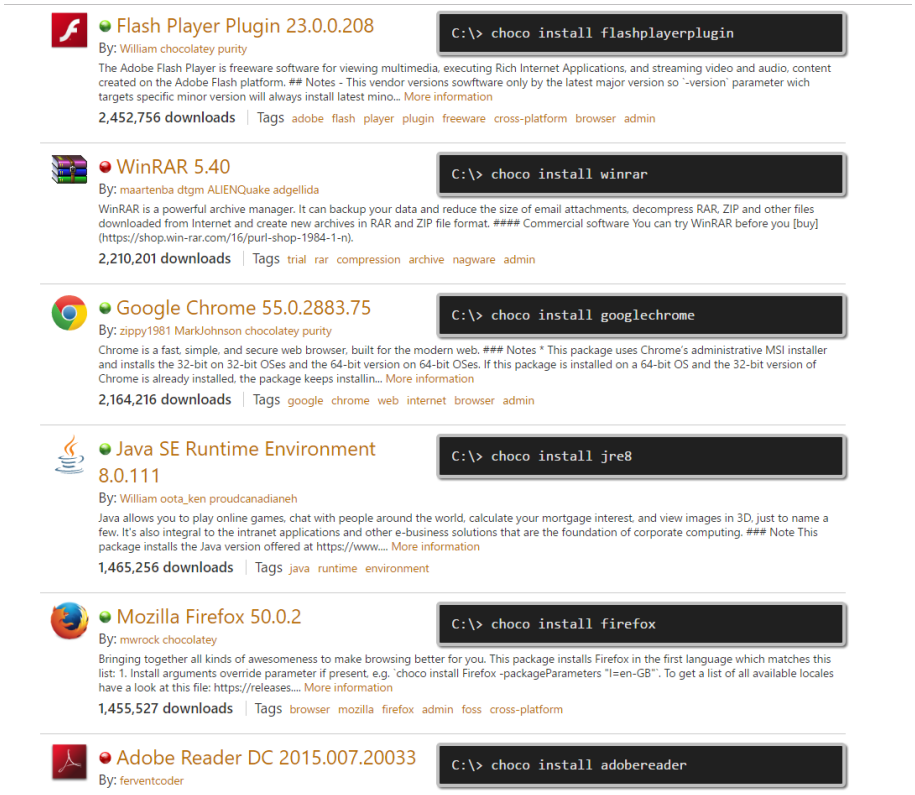
V této části příloh budou vyobrazeny postupy při implementaci zadání z Kapitoly 3.1. Budou zde využity skupiny modulů, členěných do playbooků.

A.3.1 Instalace aplikací

V této části budou vyobrazeny informace obsažené v Kapitole 3.3.1. Uživatel bude seznámen s možnostmi instalace aplikací pomocí nástroje Ansible.



Obr. A.14: Informace o modulu win_chocolatey z docs.ansible.com.



Obr. A.15: Ukázka aplikací z repozitáře Chocolatey.

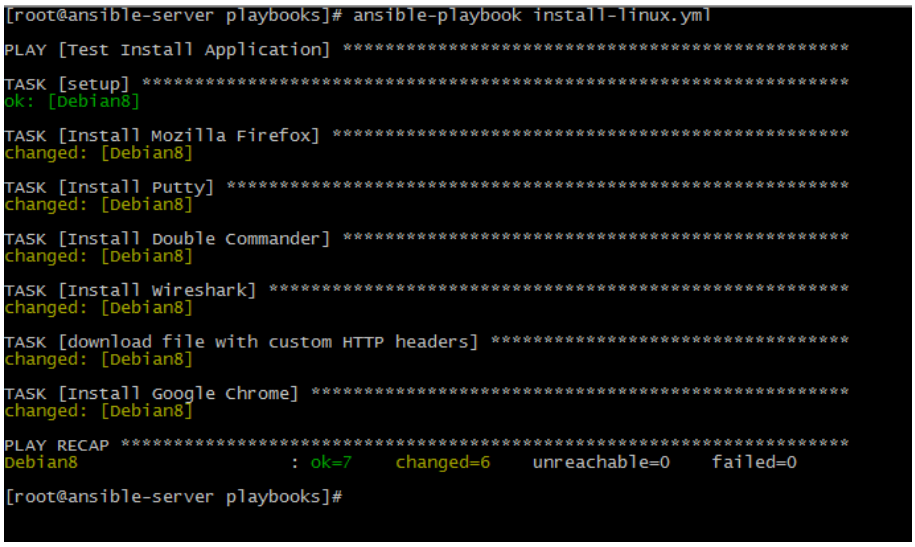
Obr. A.16: Klíče specifikující instalované 64bit aplikace na stanici.

Playbook *Install-app-windows.yml* pro instalaci aplikací na Windows.

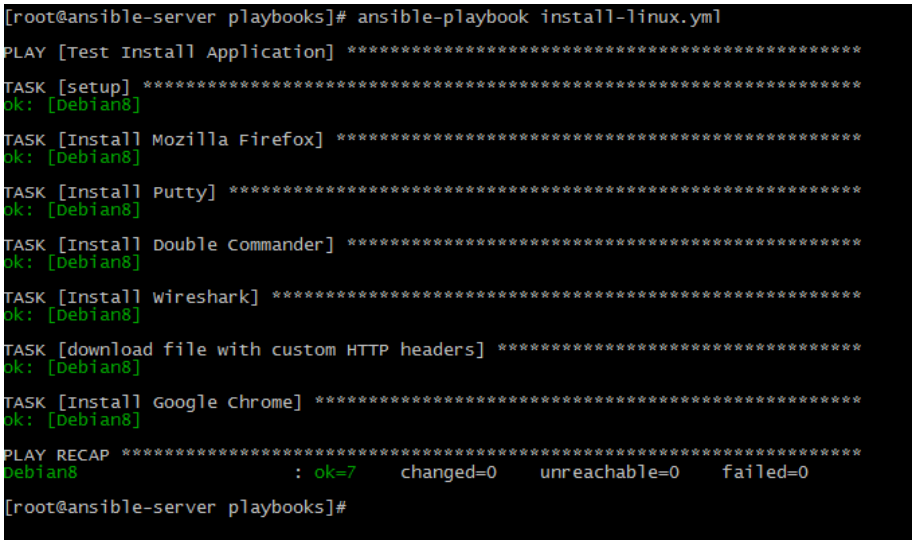
```
- name: Install applications
  hosts: windows
  tasks:
    - name: Install Mozilla Firefox
      win_chocolatey:
        name: firefox
        state: present

    - name: Install Putty
      win_chocolatey:
        name: putty
        state: present

    - name: Install Google Chrome
      win_chocolatey:
        name: googlechrome
        state: present

    - name: Install Total Commander
      win_chocolatey:
        name: totalcommander
        version: 8.01
        state: present

    - name: Install Wireshark
      win_chocolatey:
        name: wireshark
        state: present

     - name: Install Microsoft Visual Studio
                 win_package:
        path: "https://download.microsoft.com/download/0/B/C/
   0BC321A4-013F-479C-84E6-4A2F90B11269/vs_community.exe"
        product_id: "{DE064F60-6522-3310-9665-B5E3E78B3638}"
        state: present

    - name: Download the Microsoft Office 2010 installer
      win_get_url:
        url: 'http://muj-server/files/office/office2010.msi'
        dest: 'C:\Users\admin\Downloads\office2010.msi'

    - name: Install Office 2010 MSI
      win_msi:
        path: 'C:\Users\admin\Downloads\office2010.msi'
        state: present
```

Playbook *install-linux.yml* pro instalaci aplikací na Debian a Ubuntu.

```
- name: Test Install Application
  hosts: linux
  tasks:
    - name: Install Mozilla Firefox
      apt:
        name:  firefox-esr-l10n-cs
        state: present

    - name: Install Putty
      apt:
        name: putty
        state: present

    - name: Install Double Commander
      apt:
        name: doublecmd-gtk
        state: present

    - name: Install Wireshark
      apt:
        name: wireshark
        state: present

    - name: Download Google Chrome.deb
      get_url:
        url: https://dl.google.com/linux/direct/google-chrome-stable_current_amd64.deb
        dest: /home/Downloads/google-chrome-stable_current_amd64.deb

    - name: Install Google Chrome
      apt:
        deb: /home/Downloads/google-chrome-stable_current_amd64.deb
```



Obr. A.17: Výpis Ansible, po instalaci aplikací.

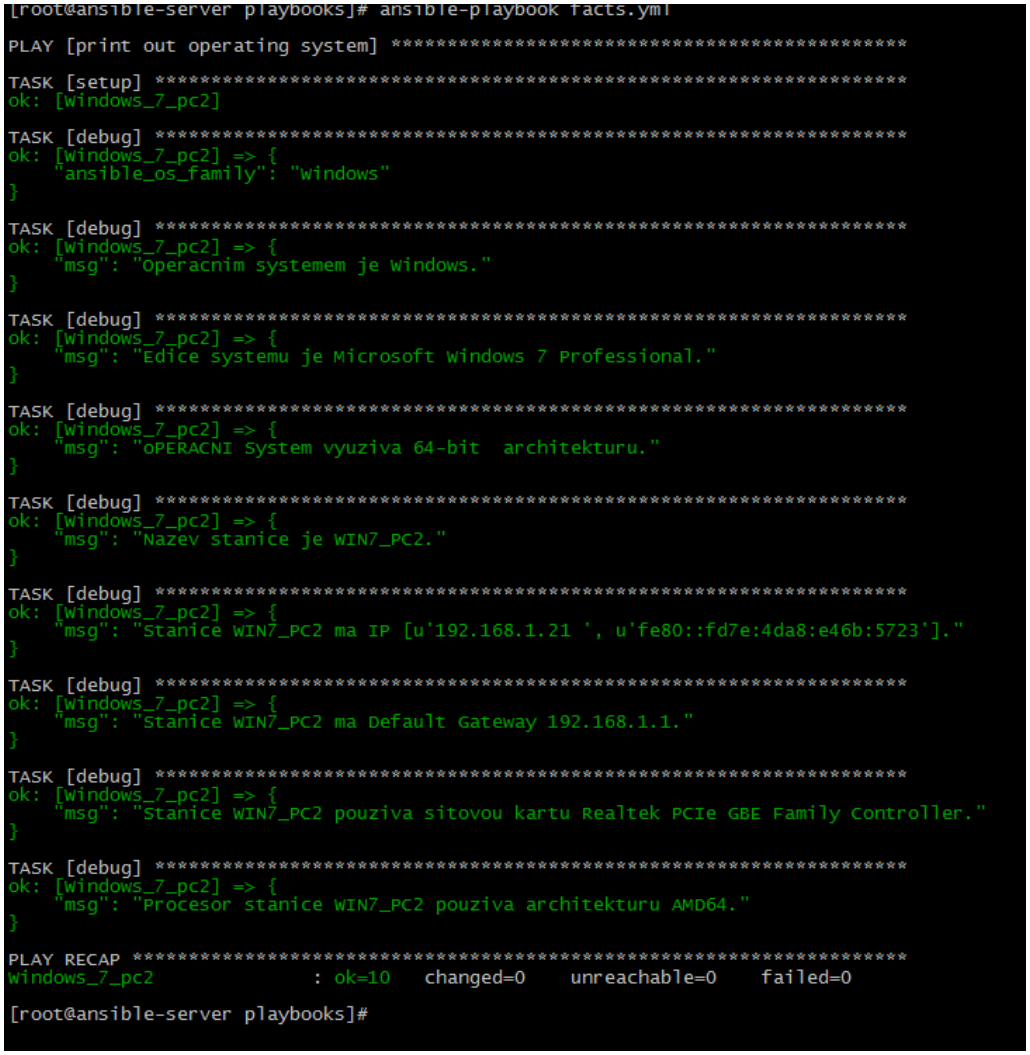


Obr. A.18: Výpis Ansible, po opětovném spuštění instalace aplikací.

A.3.2 Informace o stroji

Playbook *facts.yml* pro získání informací z Windows strojů.

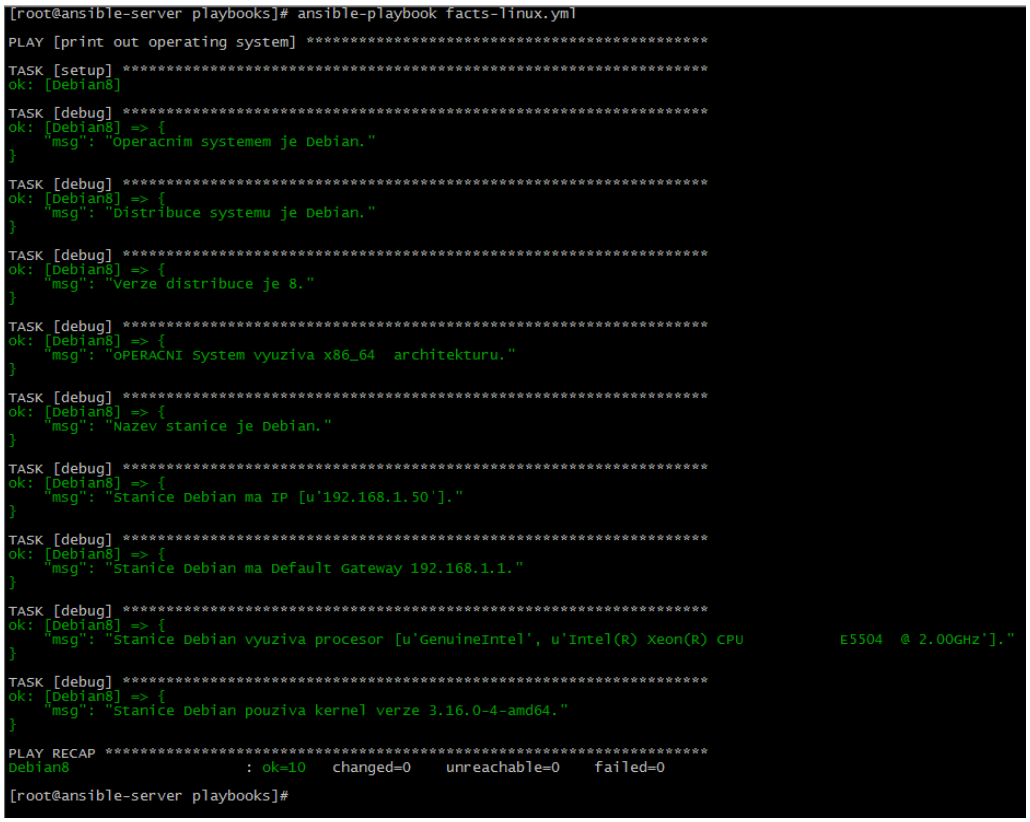
```
[root@ansible-server playbooks]# cat facts.yml
     - name: print out operating system
       hosts: windows
       gather_facts: True

       tasks:

#        - debug: var=ansible_os_family

        - debug: msg="Operacnim systemem je {{ ansible_os_family }}."

#        - debug: var=ansible_os_name

        - debug: msg="Edice systemu je {{ ansible_os_name }}."

#        - debug: var=ansible_architecture

        - debug: msg="oPERACNI System vyuziva {{ ansible_architecture }}  architekturu."

#        - debug: var=ansible_hostname

        - debug: msg="Nazev stanice je {{ ansible_hostname }}."

#        - debug: var=ansible_ip_addresses

        - debug: msg="Stanice {{ ansible_hostname }} ma IP {{ ansible_ip_addresses }}."

#        - debug: var=ansible_interfaces[0].default_gateway

        - debug: msg="Stanice {{ ansible_hostname }} ma Default Gateway
                              {{ ansible_interfaces[0].default_gateway }}."

#        - debug: var=ansible_interfaces[0].interface_name

        - debug: msg="Stanice {{ ansible_hostname }} pouziva sitovou kartu
                              {{ ansible_interfaces[0].interface_name }}."

#        - debug: var=ansible_env.PROCESSOR_ARCHITECTURE

        - debug: msg="Procesor stanice {{ ansible_hostname }} pouziva architekturu
                              {{ ansible_env.PROCESSOR_ARCHITECTURE }}."
```



Obr. A.19: Informace o stanici Windows_7_pc2, pomocí modulu debug.

Playbook *facts-linux.yml* pro získání informací z Linux strojů.

```
    - name: print out operating system
      hosts: linux
      gather_facts: True

      tasks:

#       - debug: var=ansible_os_family

       - debug: msg="Operacnim systemem je {{ ansible_os_family }}."

#       - debug: var=ansible_distribution

       - debug: msg="Distribuce systemu je {{ ansible_distribution }}."

#       - debug: var=ansible_distribution

       - debug: msg="Verze distribuce je {{ ansible_distribution_major_version }}."

#       - debug: var=ansible_architecture

       - debug: msg="oPERACNI System vyuziva {{ ansible_architecture }}  architekturu."

#       - debug: var=ansible_hostname

       - debug: msg="Nazev stanice je {{ ansible_hostname }}."

#       - debug: var=ansible_all_ipv4_addresses

       - debug: msg="Stanice {{ ansible_hostname }} ma IP {{ ansible_all_ipv4_addresses }}."

#       - debug: var=ansible_default_ipv4.gateway

       - debug: msg="Stanice {{ ansible_hostname }} ma Default Gateway
                           {{ ansible_default_ipv4.gateway }}."

#       - debug: var=ansible_processor

       - debug: msg="Stanice {{ ansible_hostname }} vyuziva procesor {{ ansible_processor }}."

#       - debug: var=ansible_kernel

       - debug: msg="Stanice {{ ansible_hostname }} pouziva kernel verze {{ ansible_kernel }}."
```



Obr. A.20: Informace o stanici Debian, pomocí modulu debug.

A.3.3 Restart a vypnutí stanice

Playbook *reboot-windows.yml* pro restartování Windows strojů.

```
[root@ansible-server playbooks]# cat reboot-windows.yml

    - name: Test reboot windows system
      hosts: windows
      tasks:

        - name: Reboot system
          win_reboot:
             connect_timeout_sec: 45
```

```
[root@ansible-server playbooks]# ansible-playbook reboot-windows.yml

PLAY [Test reboot windows system] **********************************************

TASK [setup] *******************************************************************
ok: [windows_7_pc2]

TASK [Reboot system] ***********************************************************
changed: [windows_7_pc2]

PLAY RECAP *********************************************************************
windows_7_pc2              : ok=2    changed=1    unreachable=0    failed=0

[root@ansible-server playbooks]#
```

Obr. A.21: Výpis z playbooku pro restart windows strojů.

Playbook *reboot-linux.yml* pro restartování Linux strojů.

```
[root@ansible-server playbooks]# cat reboot-linux.yml

    - name: Test reboot linux system
      hosts: linux
      tasks:

        - name: Restart machine
          shell: sleep 2 && shutdown -r now "Ansible updates triggered"
          async: 1
          poll: 0
          ignore_errors: true

        - name: Waiting for server to come back
          local_action: wait_for host={{ inventory_hostname }} state=started delay=30 timeout=60
```

```
[root@ansible-server playbooks]# ansible-playbook  reboot-linux.yml

PLAY [Test reboot linux system] ************************************************

TASK [setup] *******************************************************************
ok: [Debian8]

TASK [restart machine] *********************************************************
ok: [Debian8]

TASK [waiting for server to come back] *****************************************
ok: [Debian8 -> localhost]

PLAY RECAP *********************************************************************
Debian8                    : ok=3    changed=0    unreachable=0    failed=0
```

Obr. A.22: Výpis z playbooku pro restart linux strojů.

Playbook *shutdown-windows.yml* pro vypnutí Windows strojů.

```
[root@ansible-server playbooks]# cat shutdown-windows.yml

    - name: Test shutdown windows system
      hosts: windows
      tasks:

         - name: Shutdown machine
           win_shell: shutdown -t 10 -s
           async: 10
           poll: 0
           ignore_errors: true
```

```
[root@ansible-server playbooks]# ansible-playbook shutdown-windows.yml

PLAY [Test shutdown windows system] ********************************************

TASK [setup] *******************************************************************
ok: [Windows_7_pc2]

TASK [Shutdown machine] ********************************************************
ok: [Windows_7_pc2]

PLAY RECAP *********************************************************************
Windows_7_pc2              : ok=2    changed=0    unreachable=0    failed=0
```

Obr. A.23: Výpis z playbooku pro vypnutí windows strojů.

Po zdárném provedení příkazu bude systém do 10 sekund vypnut.

Playbook *shutdown-linux.yml* pro vypnutí Linux strojů.

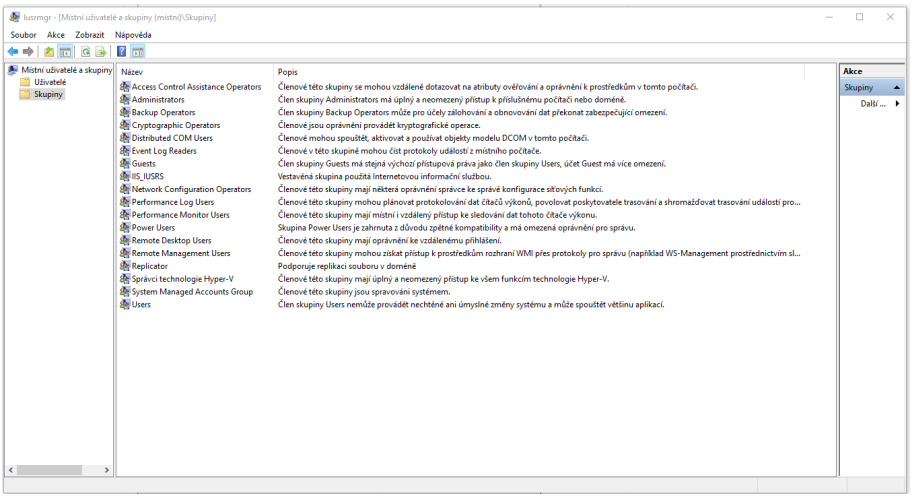
```
[root@ansible-server playbooks]# cat shutdown-linux.yml

    - name: Test shutdown linux system
      hosts: linux
      tasks:

         - name: Shutdown machine
           shell: shutdown -h
           async: 10
           poll: 0
           ignore_errors: true
```

```
[root@ansible-server playbooks]# ansible-playbook shutdown-linux.yml

PLAY [Test shutdown linux system] **********************************************

TASK [setup] *******************************************************************
ok: [Debian8]

TASK [Shutdown machine] ********************************************************
 [WARNING]: Module invocation had junk after the JSON data:   Broadcast message from root@Debian (Sat 2016-12-10 13:12:47 CST):    The system is going down for power-off at Sat 2016-12-10 13:13:47 CST!

ok: [Debian8]

PLAY RECAP *********************************************************************
Debian8                    : ok=2    changed=0    unreachable=0    failed=0
```

Obr. A.24: Výpis z playbooku pro vypnutí linux strojů.

Z Obr. A.24 je jasně patrné, že bude stanice opravdu vypnuta, dle výpisu *The system is going down...*

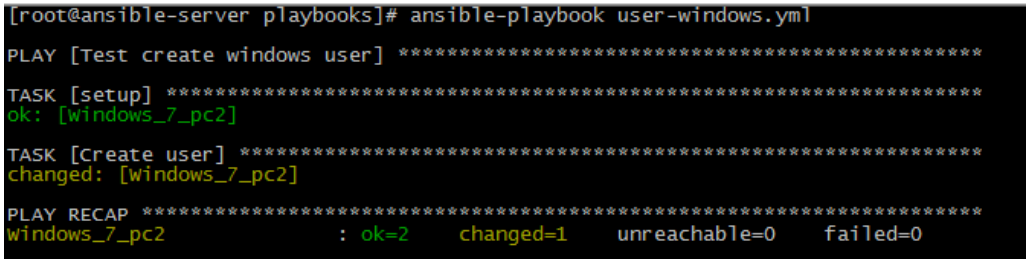
A.3.4 Tvorba uživatelů



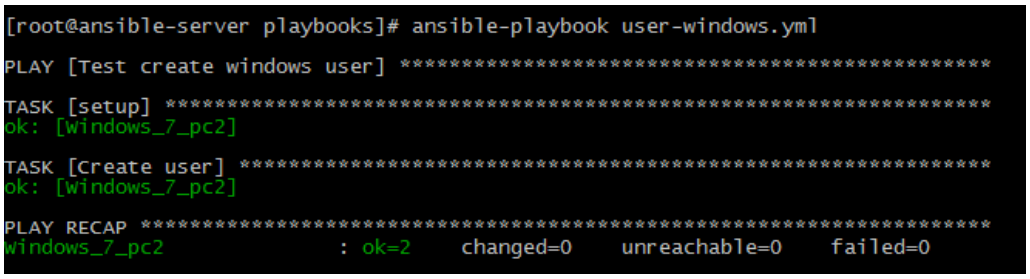
Obr. A.25: Skupiny používané ve Windows.

Playbook *user-windows.yml* pro vytvoření uživatele na Windows.

```
[root@ansible-server playbooks]# cat user-windows.yml

   - name: Test create windows user
     hosts: windows
     tasks:

        - name: Create user
          win_user:
            name: Student
            password: "Student123"
            groups: "Users"
            state: present
```



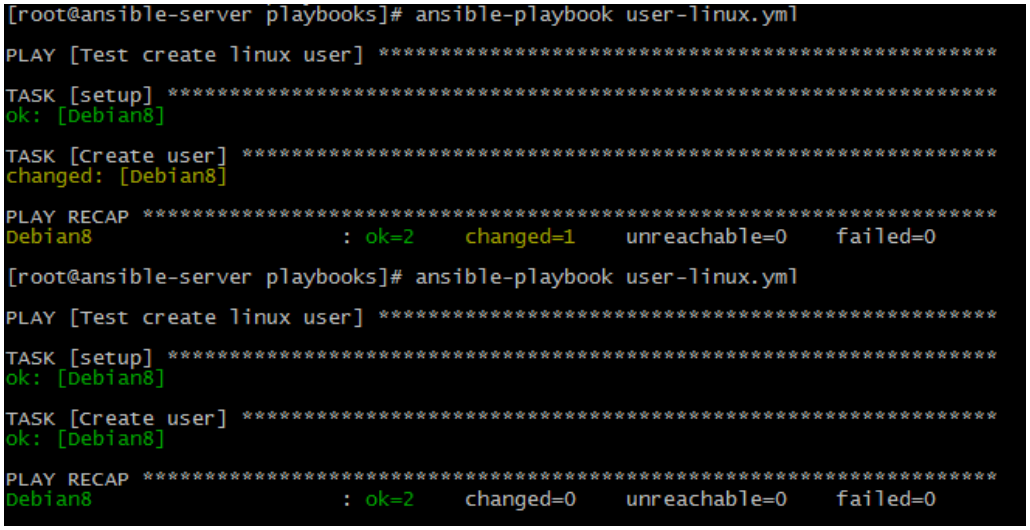
Obr. A.26: Výpis z playbooku pro vytvoření uživatele na Windows.



Obr. A.27: Výpis z playbooku pro vytvoření uživatele na Windows po opětovném spuštění.

Playbook *user-linux.yml* pro vytvoření uživatele na Linux.

```
[root@ansible-server playbooks]# cat user-linux.yml

   - name: Test create linux user
     hosts: linux
     tasks:

        - name: Create user
          user:
            name: Petr
            password: "Admin123"
            groups: "root"
            state: present
```



Obr. A.28: Výpis z playbooku pro tvorbu uživatele i s opětovným spuštěním.

A.3.5 Vytváření složek a souborů

Playbook *create-files-windows.yml* pro vytvoření složky a textového souboru spolu s jeho editací na Windows.

```
[root@ansible-server playbooks]# cat create-file-windows.yml

   - name: Test create folder, file and edit file
     hosts: windows
     tasks:

        - name: Create  directory TextFiles
          win_file:
            path: C:\TextFiles
            state: directory

        - name: Create text file in directory TextFiles
          win_file:
           path: C:\TextFiles\text.txt
           state: touch

        - name: Write text to file
          win_lineinfile:
           dest: C:\TextFiles\text.txt
           line: Ahoj Svete :-)
```

```
[root@ansible-server playbooks]# ansible-playbook create-file-windows.yml

PLAY [Test create folder, file and edit file] **********************************

TASK [setup] *******************************************************************
ok: [Windows_7_pc2]

TASK [Create  directory TextFiles] *********************************************
 [WARNING]: Module invocation had junk after the JSON data:

changed: [Windows_7_pc2]

TASK [Create text file in directory TextFiles] *********************************
changed: [Windows_7_pc2]

TASK [Write text to file] ******************************************************
changed: [Windows_7_pc2]

PLAY RECAP *********************************************************************
Windows_7_pc2              : ok=4    changed=3    unreachable=0    failed=0

[root@ansible-server playbooks]# ansible-playbook create-file-windows.yml

PLAY [Test create folder, file and edit file] **********************************

TASK [setup] *******************************************************************
ok: [Windows_7_pc2]

TASK [Create  directory TextFiles] *********************************************
ok: [Windows_7_pc2]

TASK [Create text file in directory TextFiles] *********************************
changed: [Windows_7_pc2]

TASK [Write text to file] ******************************************************
ok: [Windows_7_pc2]

PLAY RECAP *********************************************************************
Windows_7_pc2              : ok=4    changed=1    unreachable=0    failed=0
```

Obr. A.29: Výpis z playbooku pro vytvoření složky,souboru spolu s jeho editací na Windows.

Playbook *create-file-linux.yml* pro vytvoření složky a textového souboru spolu s jeho editací na Linux.

```
[root@ansible-server playbooks]# cat create-file-linux.yml

   - name: Test create folder, file and edit file
     hosts: linux
     tasks:

        - name: Create  directory TextFiles
          file:
            path: /home/debian/TextFiles
            state: directory

        - name: Create text file in directory TextFiles
          file:
           path: /home/debian/TextFiles/text.txt
           state: touch

        - name: Write text to file
          lineinfile:
           dest: /home/debian/TextFiles/text.txt
           line: Ahoj Svete :-)
```

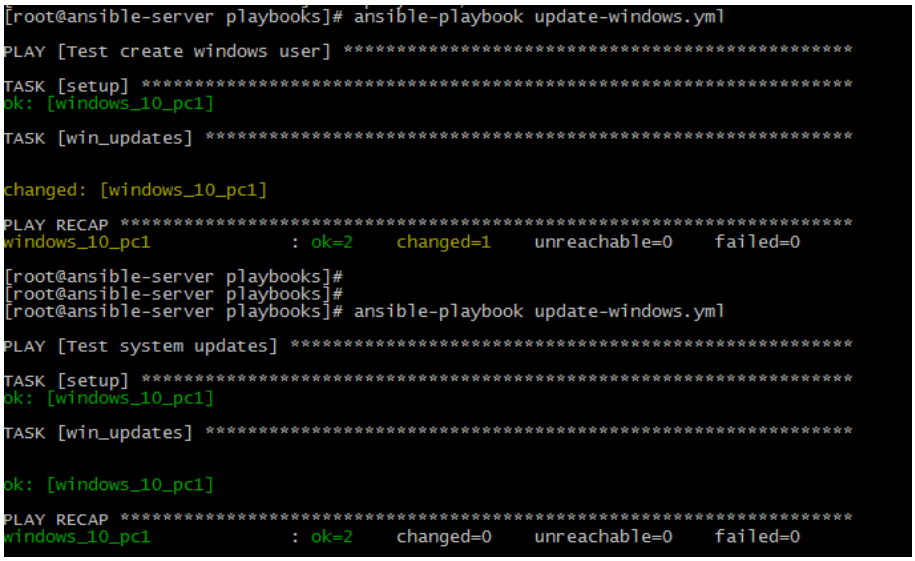
```
[root@ansible-server playbooks]# ansible-playbook create-file-linux.yml

PLAY [Test create folder, file and edit file] **********************************

TASK [setup] *******************************************************************
ok: [Debian8]

TASK [Create  directory TextFiles] *********************************************
changed: [Debian8]

TASK [Create text file in directory TextFiles] *********************************
changed: [Debian8]

TASK [Write text to file] ******************************************************
changed: [Debian8]

PLAY RECAP *********************************************************************
Debian8                    : ok=4    changed=3    unreachable=0    failed=0

[root@ansible-server playbooks]# ansible-playbook create-file-linux.yml

PLAY [Test create folder, file and edit file] **********************************

TASK [setup] *******************************************************************
ok: [Debian8]

TASK [Create  directory TextFiles] *********************************************
ok: [Debian8]

TASK [Create text file in directory TextFiles] *********************************
changed: [Debian8]

TASK [Write text to file] ******************************************************
ok: [Debian8]

PLAY RECAP *********************************************************************
Debian8                    : ok=4    changed=1    unreachable=0    failed=0
```

Obr. A.30: Výpis z playbooku pro vytvoření složky,souboru spolu s jeho editací na Linux.

A.3.6 Instalace aktualizací

Playbook *update-windows.yml* pro aktualizaci systému Windows.

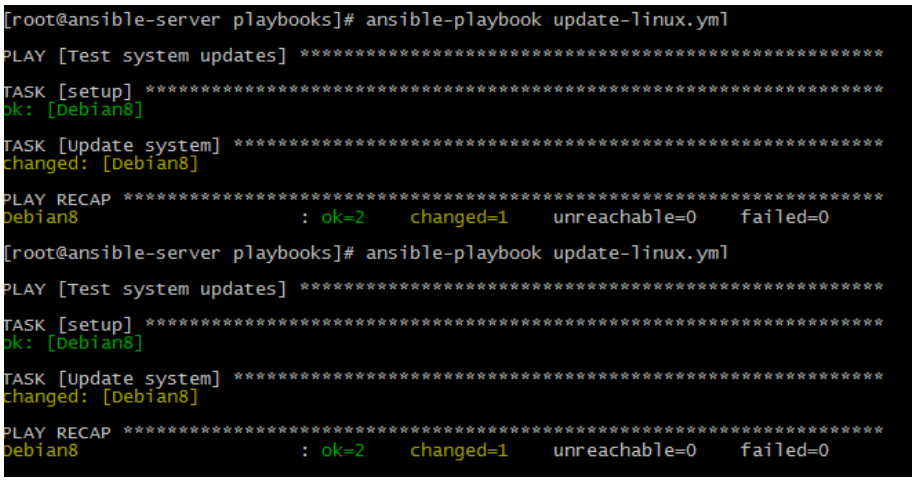
```
[root@ansible-server playbooks]# cat update-windows.yml

   - name: Test system updates
     hosts: windows
     tasks:

        - win_updates:
            category_names:
               - Updates

            state: installed
```



Obr. A.31: Výpis z playbooku pro aktualizaci systému Windows.

Playbook *update-linux.yml* pro aktualizaci systému Linux.

```
[root@ansible-server playbooks]# cat update-linux.yml

   - name: Test system updates
     hosts: linux
     tasks:

       - name: Update system
         apt:
           update_cache: yes
```



Obr. A.32: Výpis z playbooku pro aktualizaci systému Linux.

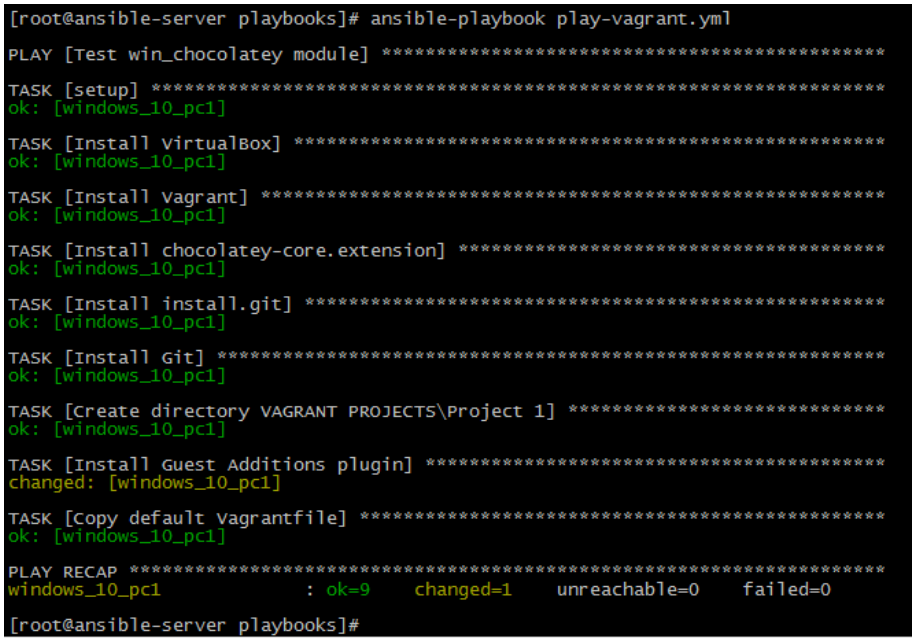
A.3.7 Vagrant pomocí Ansible

Playbook *play-vagrant.yml* pro instalaci a konfiguraci nástroje Vagrant.

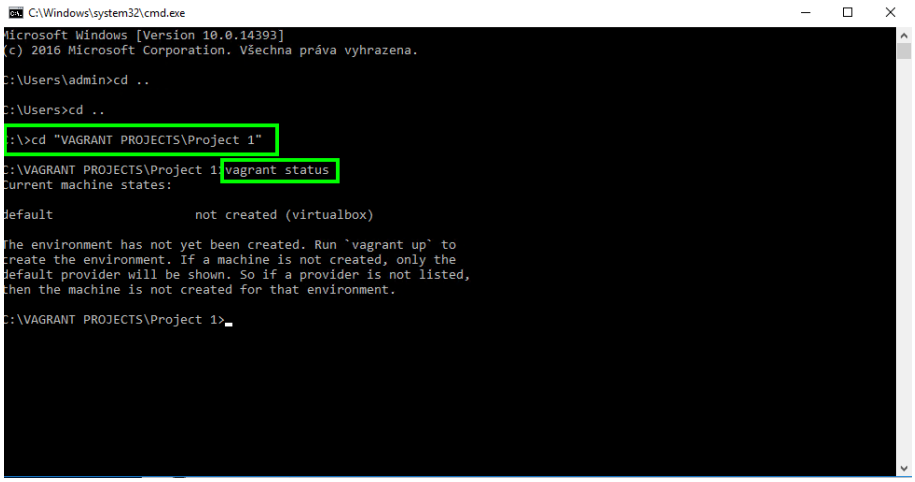
```
[root@ansible-server playbooks]# cat play-vagrant.yml
- name: Test win_chocolatey module
  hosts: windows
  tasks:
    - name: Install VirtualBox
      win_chocolatey:
        name: virtualbox
        state: present
    - name: Install Vagrant
      win_chocolatey:
        name: vagrant
        state: present
    - name: Install chocolatey-core.extension
      win_chocolatey:
        name: chocolatey-core.extension
        state: present
    - name: Install install.git
      win_chocolatey:
        name: git.install
        state: present
    - name: Install Git
      win_chocolatey:
        name: git
        state: present
    - name: Create directory VAGRANT PROJECTS\Project 1
      win_file:
         path: 'C:\VAGRANT PROJECTS\Project 1'
         state: directory
    - name: Install Guest Additions plugin
      raw: vagrant plugin install vagrant-vbguest
    - name: Copy default Vagrantfile
      win_copy:
        src: /etc/ansible/files/files/Vagrantfile
        dest: 'C:\VAGRANT PROJECTS\Project 1\'
```

Obsah souboru *Vagrantfile*.

```
Vagrant.configure("2") do |config|
config.vm.box = "centos/7"
config.vm.network "forwarded_port", guest: 80, host: 8080
config.vm.synced_folder ".", "/vagrant", type: "virtualbox"
end
```



Obr. A.33: Výpis z playbooku pro instalaci nástroje Vagrant.



Obr. A.34: Ověření funkčnosti nástroje Vagrant, výpisem status.

A.4 Tvorba webové aplikace

V Kapitole 4 byly probrány základní informace pro tvorbu webové aplikace z teoretické části. V této části budou prakticky probrány a názorně předvedeny potřebné nástroje spolu s jejich konfigurací. Tak jako v teoretické části, bude i praktická část rozdělena na potřebné nástroje pro *Vývoj* a *Provoz*.

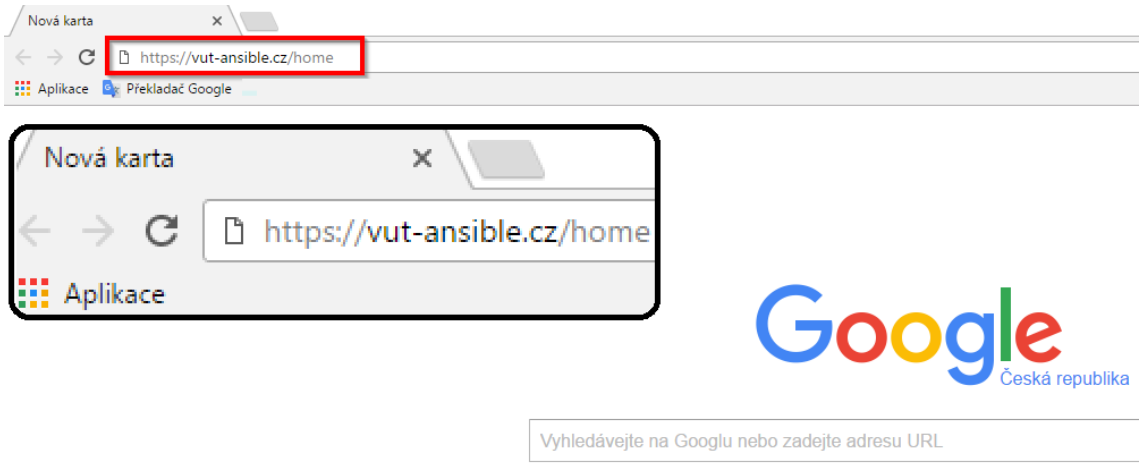
A.4.1 Vývoj

Teorie a seznámení s nástroji pro vývoj webové aplikace byly předvedeny, v následujícím textu bude probrána instalace, konfigurace a použití nástrojů jako je *JDK* (Java Development Kit) 4.2.1, *Maven* 4.2.1 a *STS* (Spring Tool Suit) 4.2.1. Všechny tyto nástroje jsou potřebné pro práci s již zmíněným frameworkem *Spring Web MVC* 4.2.1.

Spring Web MVC Framework

Na Obr. 4.3 byla vyobrazena komunikační architektura Spring MVC modelu. Pro lepší objasnění komunikace uvnitř této architektury, bude přiložena obrázková dokumentace.

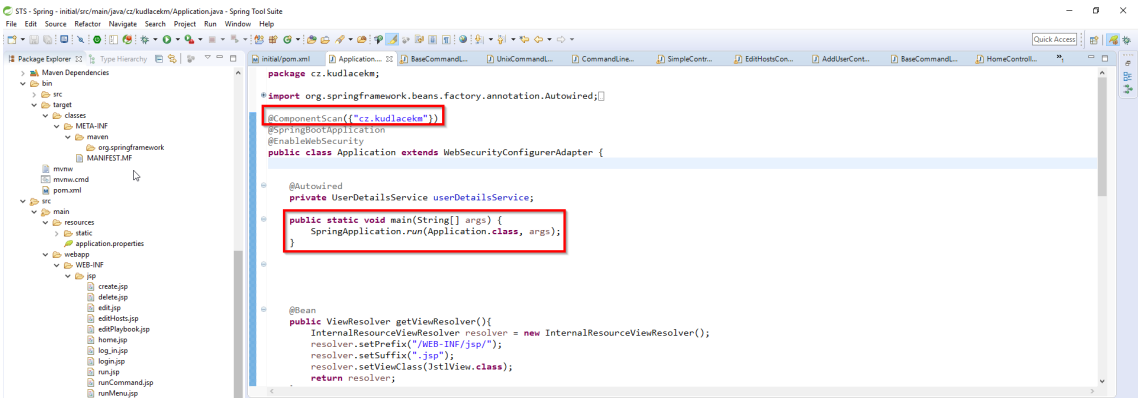
1. **Žádám o načtení stránky vut-ansible.cz/home pomocí HTTP žádosti metodou GET.**



Obr. A.35: Požadavek na stránku vut-ansible.cz/home.

2. **Kde se nachází controller pro stránku /home volaný pomocí HTTP žádosti metodou GET?**

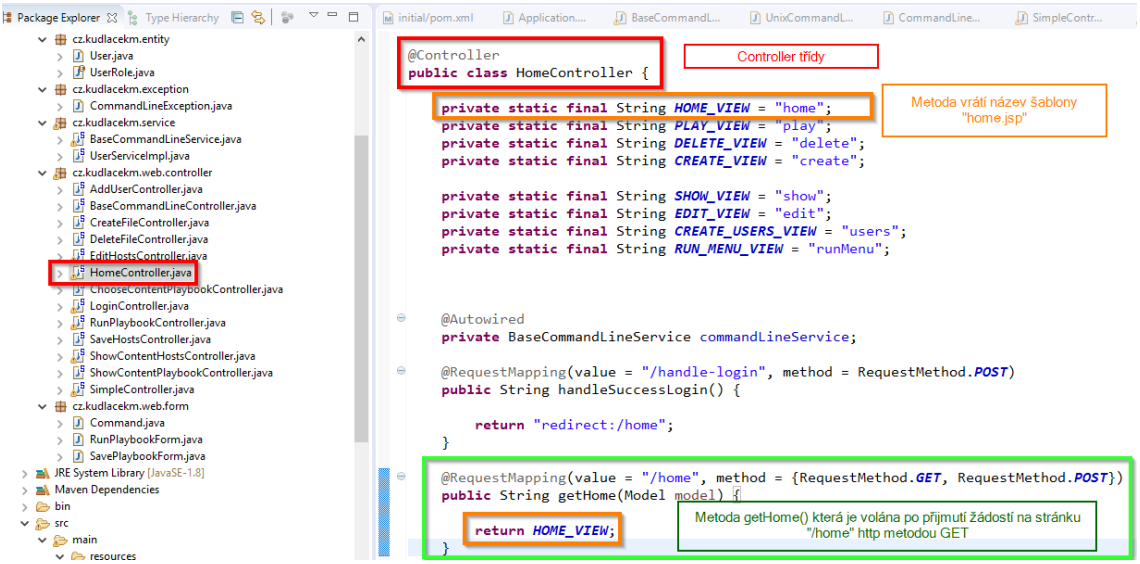
Spring aplikace hledá v celém projektu controller, který je určen pro HTTP dotaz metodou GET, na stránku /home. Pomocí anotace *@ComponentScan* je Spring schopen nalézt odpovídající controller obsažený v projektu tak, že projde všechny třídy, které jsou o anotované anotací *@Controller*. V této třídě se pak snaží nalézt metodu, která splňuje potřebné požadavky jako adresa /home a žádost metodou GET.



Obr. A.36: Hledání controlleru pro dotaz na stránku /home.

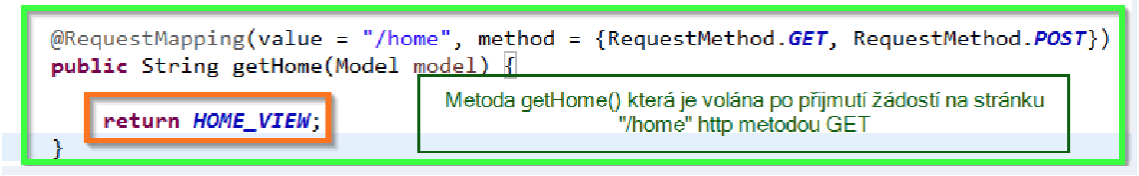
3. **Dotaz na stránku /home pomocí metody GET zajišťuje controller HomeController.java Obr. A.37**

4. **Zpřístupni data dostupná v controlleru HomeController.java.**



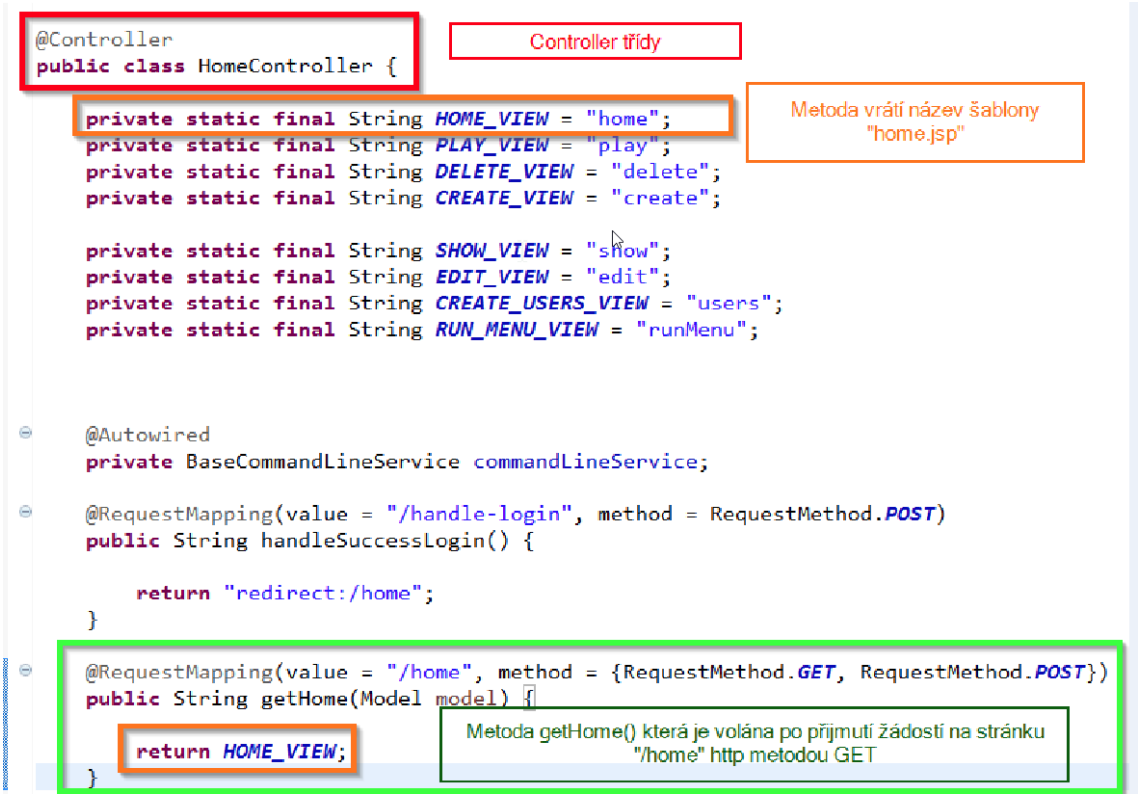
Obr. A.37: Obsah controlleru HomeController.java

5. **Zpřístupni data obsažená v metodě getHome(), která je obsažena v controlleru HomeController.java.** A.38



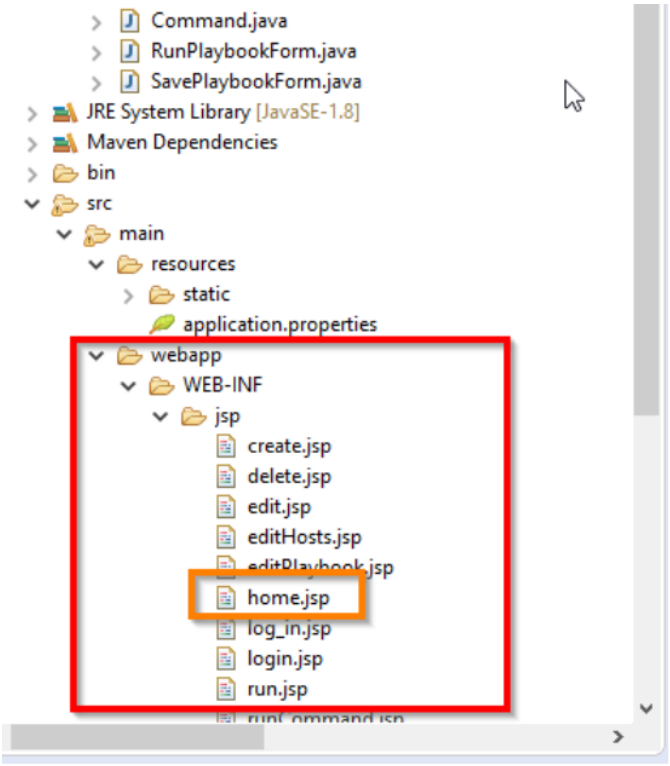
Obr. A.38: Volání metody *getHome()*.

6. **Metoda vrací tyto data spolu s názvem grafické šablony home.jsp.** A.39



Obr. A.39: Data vrácená metodou *getHome()*.

7. **V HomeController.java byla obsažena tato data spolu s názvem šablony home.jsp.**
8. **Kde se nachází šablona home.jsp?**
9. **Šablona se nachází v WEB-INF\jsp\.** A.40
10. **Zpřístupni data šablony home.jsp.** A.41
11. **Zde jsou data šablony home.jsp.** A.42
12. **Zde je stránka s grafickým vzhledem šablony home.jsp a daty z metody getHome().** A.43



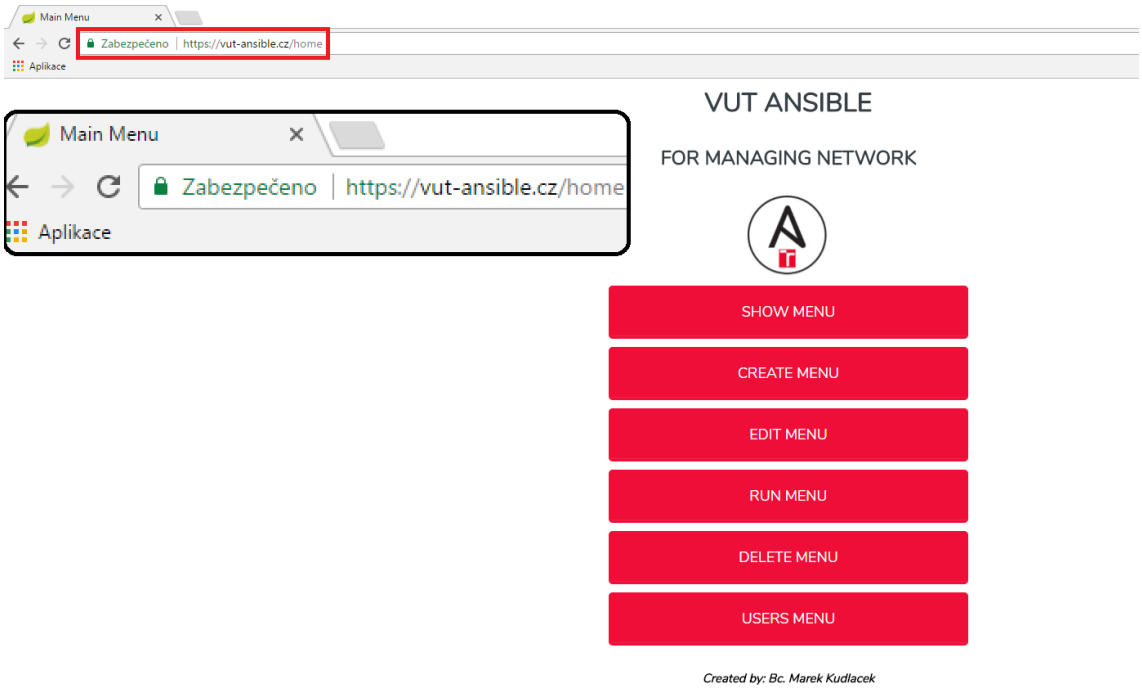
Obr. A.40: Umístění šablony home.jsp.

```java
package cz.kudlacekm;

import org.springframework.beans.factory.annotation.Autowired;

@ComponentScan({"cz.kudlacekm"})
@SpringBootApplication
@EnableWebSecurity
public class Application extends WebSecurityConfigurerAdapter {

    @Autowired
    private UserDetailsService userDetailsService;

    public static void main(String[] args) {
        SpringApplication.run(Application.class, args);
    }

    @Bean
    public ViewResolver getViewResolver(){
        InternalResourceViewResolver resolver = new InternalResourceViewResolver();
        resolver.setPrefix("/WEB-INF/jsp/");
        resolver.setSuffix(".jsp");
        resolver.setViewClass(JstlView.class);
        return resolver;
    }
```

Obr. A.41: Metoda *getViewResolver()* vracející data šablony home.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@ taglib prefix="spring" uri="http://www.springframework.org/tags"%>
<%@ taglib prefix="form" uri="http://www.springframework.org/tags/form"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>

<head>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Main Menu</title>
<link rel="stylesheet" href="css/normalize.css">
<link href='https://fonts.googleapis.com/css?family=Nunito:400,300'
    rel='stylesheet' type='text/css'>
<link rel="stylesheet" href="/data/css/login.css">
</head>
<body>


    <h1>VUT ANSIBLE</h1>
    <h2>FOR MANAGING NETWORK</h2>

    <img src="data/img/ansible2.jpg" alt="www.vut-ansible.cz" width="100" height="100" style="vertical-align: middle" >

    <a class="button" href="/delete-menu">DELETE MENU</a>
    <a class="button" href="/create-menu">CREATE MENU</a>
    <a class="button" href="/edit-menu">EDIT MENU</a>
    <a class="button" href="/show-menu">SHOW MENU</a>
    <a class="button" href="/run-menu">RUN MENU</a>
    <a class="button" href="/users-menu">USERS MENU</a>

    <h4>Created by: Bc. Marek Kudlacek</h4>

</body>
</html>
```

Obr. A.42: Obsah dat šablony home.jsp.

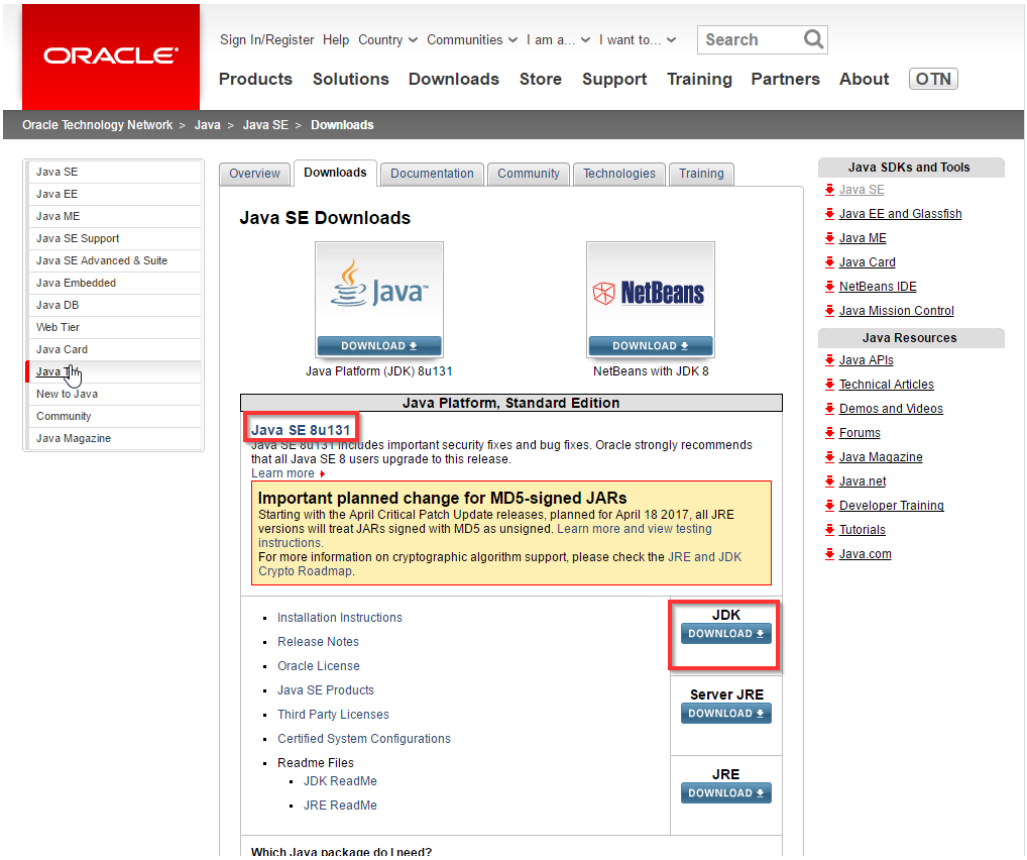


Obr. A.43: Odpověď od serveru na stránku vut-ansible.cz/home.

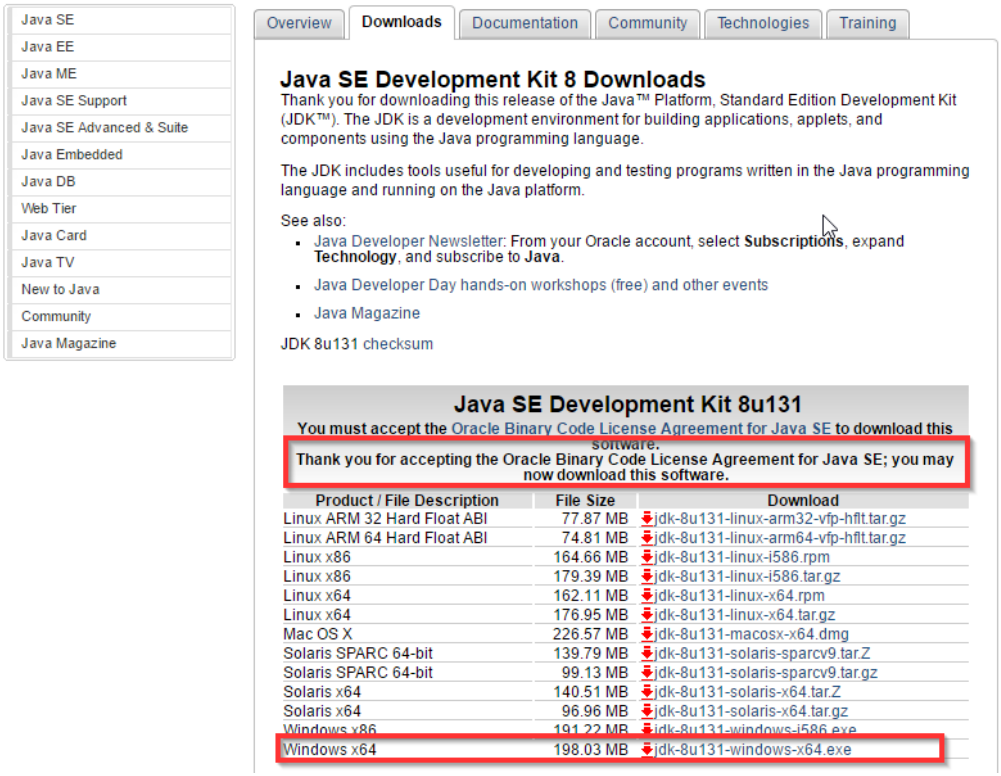
JDK (Java Development Kit)

Kde najít a jak nainstalovat *JDK* bude předvedeno v několika krátkých krocích. Instalační soubor bude stažen pro operační systém Windows 10 s 64bitovým operačním systémem, ve verzi 1.8.0. Instalace je provedena pomocí grafického průvodce a není potřeba v ní dělat nějaké zásadní změny.

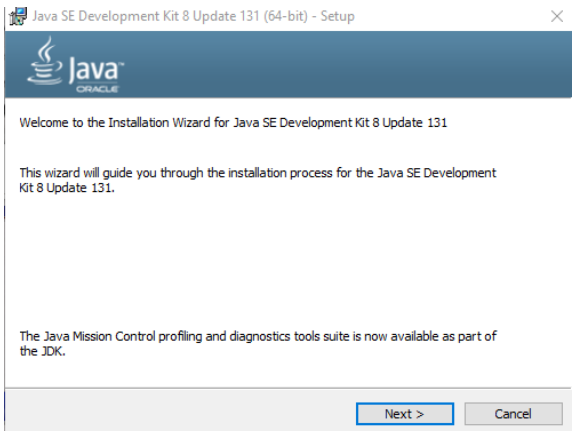
Jelikož bude webová aplikace postavena na Spring frameworku, není tedy potřeba zavádět cestu k souborům Javy do proměnné *PATH*, jelikož překlad zdrojových souborů bude probíhat ve vývojovém prostředí *STS (Spring Tool Suit)*. Bude však ale muset být vytvořena proměnná *JAVA_HOME*, kterou standardně využívají jiné programy napsané v Javě. V tomto případě to bude nástroj Maven, o kterém bude pojednáno později.



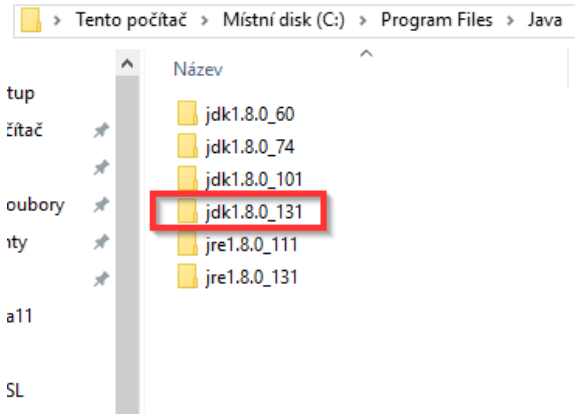
Obr. A.44: Výběr Java SE JDK 1.8.0 souboru.



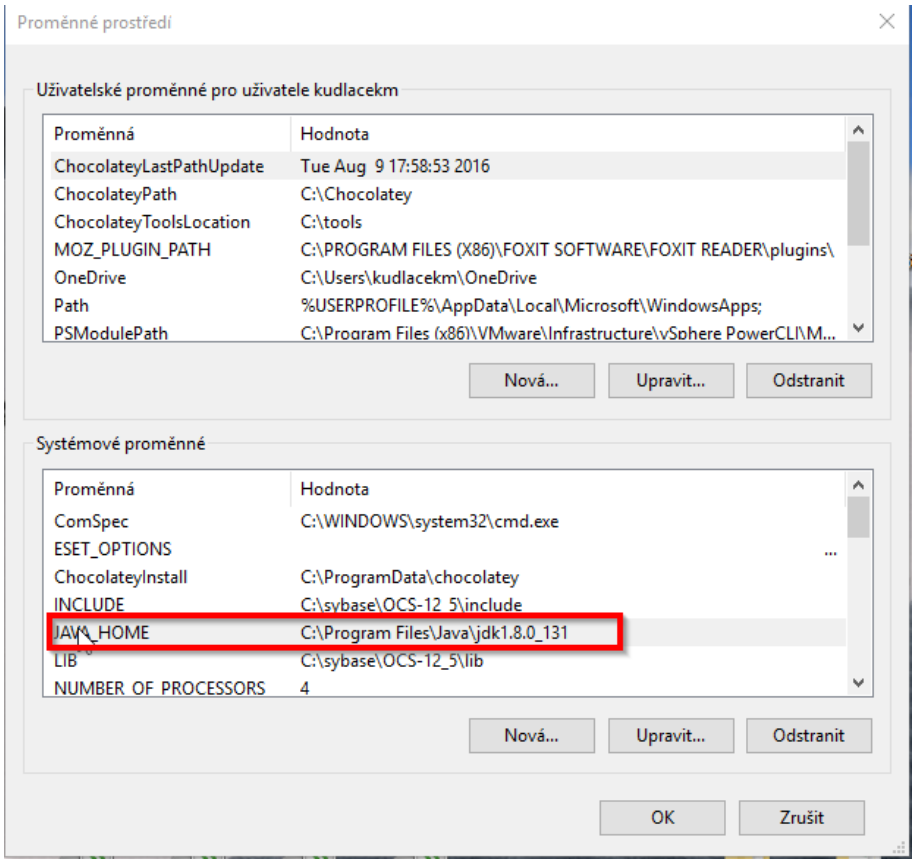
Obr. A.45: Výběr instalačního souboru pro specifický systém.



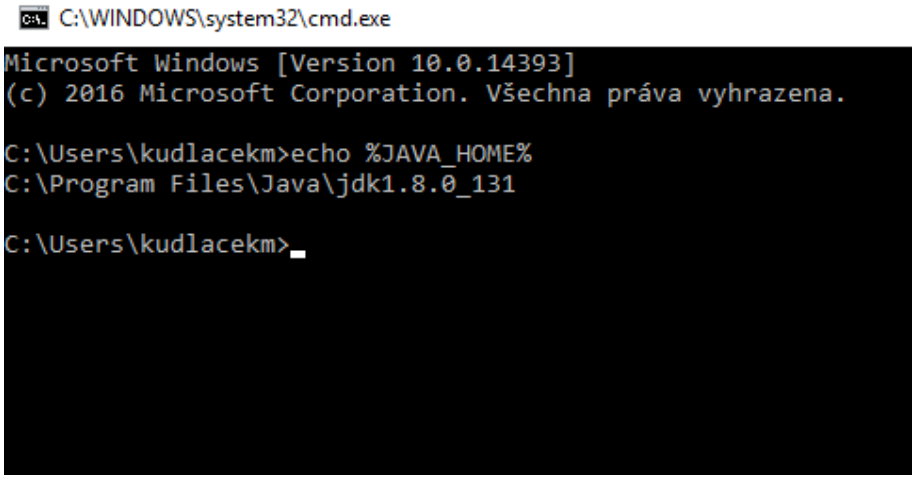
Obr. A.46: Průvodce instalací JDK 1.8.0 pro Windows.



Obr. A.47: Umístění nainstalovaných souborů JDK 1.8.0.



Obr. A.48: Vytvoření proměnné *JAVA_HOME* do proměnného prostředí systému.

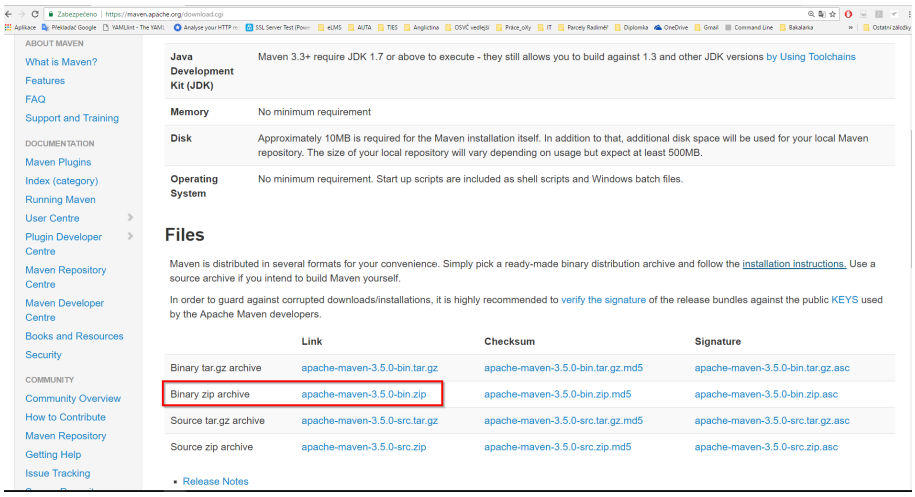


Obr. A.49: Kontrola zavedení proměnné *JAVA_HOME* do proměnného prostředí systému.

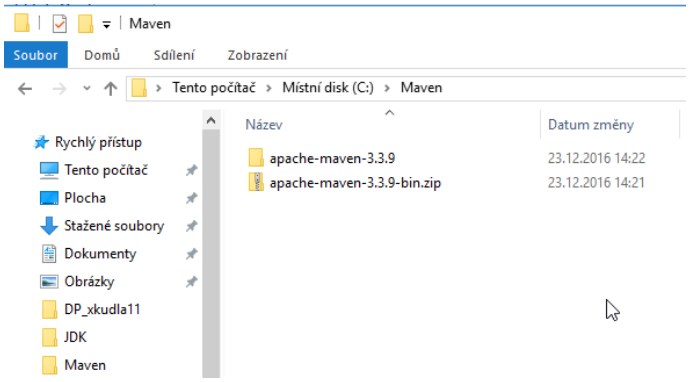
Maven

Apache Maven nástroj od společnosti *Apache Software Foundation*. Pro verzi 3.3 a vyšší je nutné mít nainstalován Java Development Kit minimálně verze 1.7, nejlépe již 1.8. Maven nevyužívá žádné .exe soubory pro instalaci, stačí pouze stažený soubor rozbalit na disk. Nakonec musí být do systémové proměnné *PATH* zavedena cesta k rozbaleným Maven souborům, aby mohly být příkazy spouštěny z příkazové řádky. Maven pro svou práci vyžaduje zavedení systémové proměnné *JAVA_HOME*, která již byla zavedena při instalaci *JDK* A.4.1, jak je znázorněno na Obr. A.48.

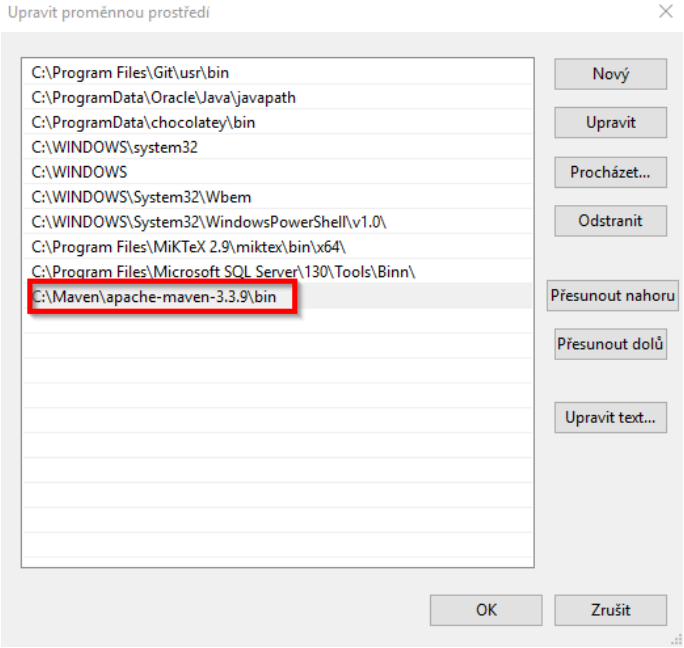
Níže bude uveden krátký grafický průvodce pro nasazení nástroje Maven pro 64bitový operační systém Windows 10. Informace jak postupovat při konfiguraci jsou dostupné i na stránce *maven.apache.org*.



Obr. A.50: Stažení konfiguračního balíku Maven, pro 64bitový systém Windows 10.

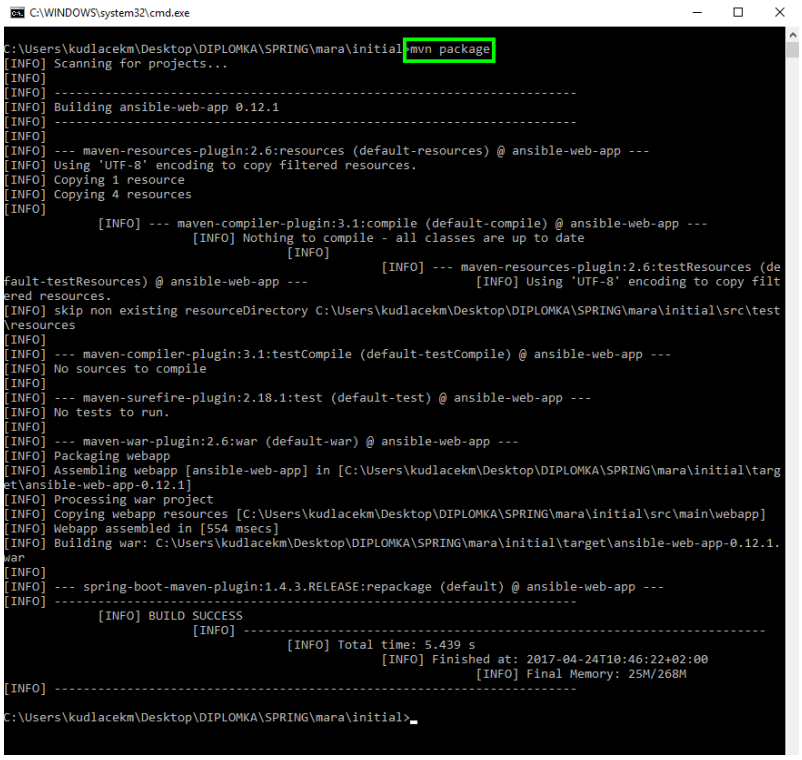


Obr. A.51: Rozbalení Maven souboru do složky.

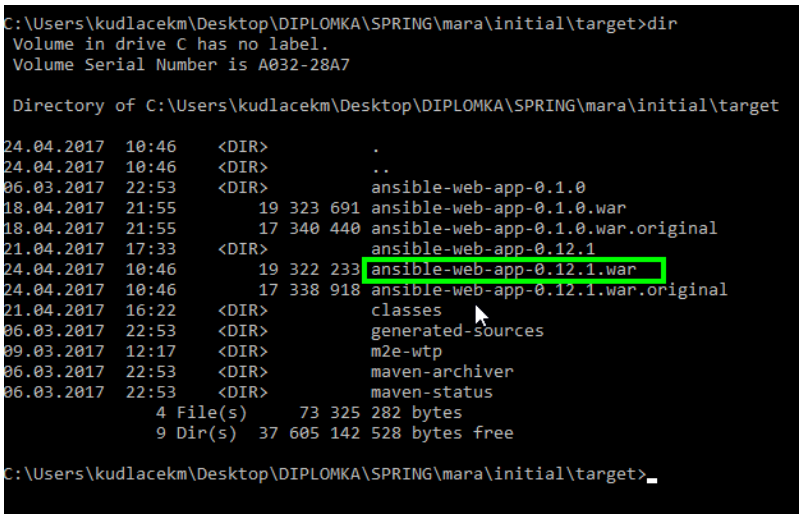


Obr. A.52: Nastavení cesty k Maven souborům, do systémové proměnné *PATH*.

V tomto textu byly představeny základní úkony potřebné pro práci s nástrojem Maven. Pro bližší a podrobnější informace, jak pracovat s nástrojem Maven a frameworkem Spring přejděte na stránku [44].



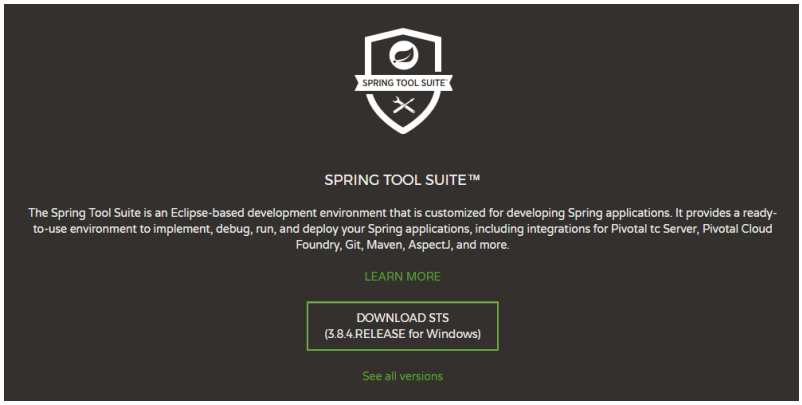
Obr. A.53: Spuštění buildu projektu pomocí *mvn package*.



Obr. A.54: Sestavený *.war* soubor se spustitelnou webovou aplikací.

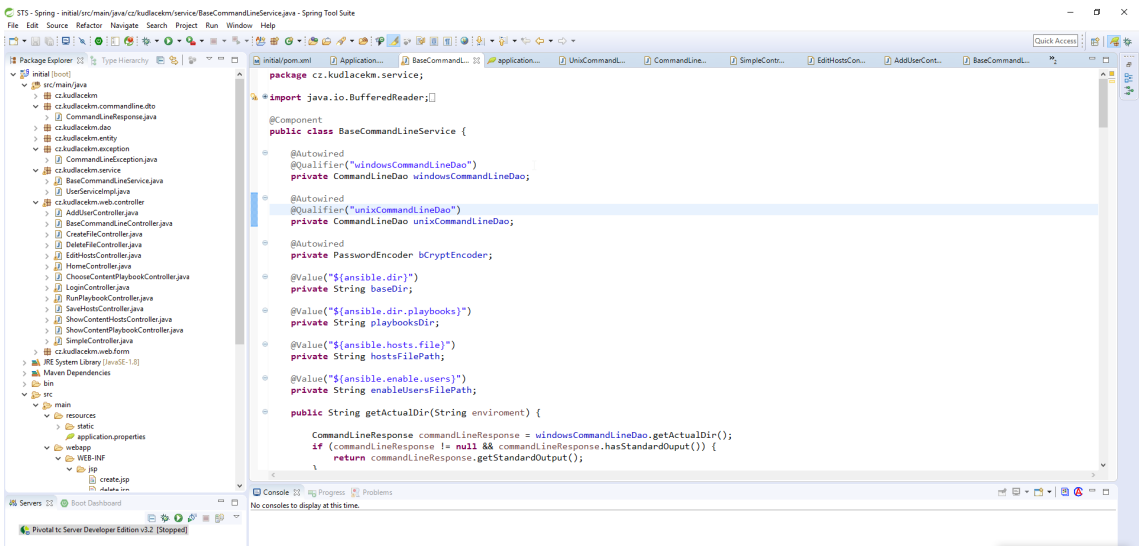
STS (Spring Tool Suit)

Vývojové prostředí postavené na základu Eclipse. Nevyžaduje zdlouhavé instalace a konfigurace, ale postačí si s pouhým rozbalením stažených souborů na disk. Poté již stačí spustit *STS.exe* a započít vývoj Spring aplikace. Na stránce [45] je dostupný soubor ke stažení. Vytvořená aplikace je psána ve verzi 3.8.3. Níže však je k vidění již dostupná verze 3.8.4.



Obr. A.55: Stažení balíku Spring Tool Suit.

Po rozbalení a spuštění prostředí jsou na první pohled vidět stejné rysy jako v Eclipse.



Obr. A.56: Vývojové prostředí Spring Tool Suit.

V tuto chvíli je Spring Tool Suit připraven.

A.4.2 Provoz

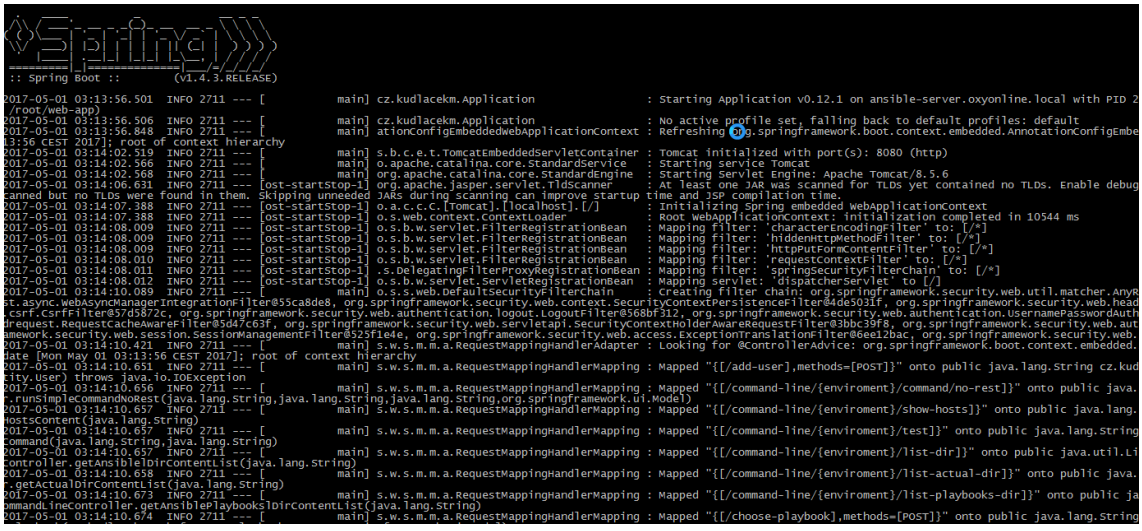
Kapitola obsahuje souhrn použitých nástrojů, potřebných pro spuštění a provoz webové aplikace. Seznamuje s postupem instalací, konfigurací a použitých příkazů. Vše je doplněno grafickou dokumentací.

JRE

Java-1.8.0-openjdk je instalována z *Red Hat* repozitáře, který je výchozím *rpm* repozitářem pro distribuci CentOS. Ihned po nainstalování je systém schopen spouštět aplikace psané v jazyce Java.

```
nohup java -jar ansible-web-app-0.12.1.war > /root/web-app/log.txt 2>&1
```

Aplikace *ansible-web-app-0.12.1.*, bude spuštěna díky příkazu *java* s parametrem *-jar*. Aby při odhlášení a odpojení serverové relace nedošlo k ukončení běhu aplikace signálem *SIGNHUP*, tak je použit příkaz *nohup*, který spustí webovou aplikaci na pozadí systému. *2>& 1* směruje standardní chybový výstup do standardního výstupu, který je ukládán do souboru log.txt. Do souboru vypisuje všechny informace související se startováním springové aplikace. Průběh spouštění aplikace pomocí *tail -f /root/web-app/log.txt*.



Obr. A.57: Spuštění webové aplikace na serveru.

Zdárně spuštěná aplikace vypíše hlášení *Started Application in*, kde za *in* se nachází čas zahrnující dobu spouštění aplikace. Pro ověření běhu aplikace je použit příkaz

```
netstat -ntl
```

zobrazující všechny naslouchající porty. Jelikož je aplikace vytvořena pomocí Spring frameworku, ve kterém je ve výchozím nastavení používán *embedded Tomcat*, tak je použit výchozí port 8080. Ve výpisu je již vidět, že systém opravdu naslouchá na portu 8080.

```
[root@ansible-server ~]# netstat -ntl
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 0.0.0.0:111             0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:25            0.0.0.0:*               LISTEN
tcp6       0      0 :::111                  :::*                    LISTEN
tcp6       0      0 :::8080                 :::*                    LISTEN
tcp6       0      0 :::80                   :::*                    LISTEN
tcp6       0      0 :::22                   :::*                    LISTEN
tcp6       0      0 ::1:25                  :::*                    LISTEN
tcp6       0      0 :::443                  :::*                    LISTEN
[root@ansible-server ~]#
```

Obr. A.58: Výpis naslouchajících portu v systému.

Z výpisu je dále patrné, že aplikace již naslouchá i na portech 80 a 443, které jsou využity pro přístup z webových prohlížečů uživatelů. Nastavení a význam těchto portů bude probrán v Kapitole 4.2.2. Následujícím příkazem je ověřeno zdárné spuštění aplikace.

```
ps -ef | grep java
```

Příkaz vyfiltruje z běžících procesů právě proces *java*. Na Obr. A.59 je vidět běžící proces s názvem webové aplikace.

```
[root@ansible-server ~]# ps -ef | grep java
root      2742     1  0 May01 ?        00:03:03 java -jar ansible-web-app-0.12.1.war
root      4874  4840  0 15:06 pts/1    00:00:00 grep --color=auto java
[root@ansible-server ~]#
```

Obr. A.59: Výpis běžícího procesu javy.

Apache

Teoretické seznámení s webovým serverem Apache bylo probráno v Kapitole 4.2.2. Tato část textu je věnována praktickým ukázkám jak nainstalovat, nakonfigurovat a používat webový server Apache ve verzi 2.4.6. Bude zde předvedena komunikace pomocí protokolu *HTTP* a jeho nedostatky v bezpečnosti. Bude zde demonstrováno odposlechnutí přihlašovacích údajů uživatele, který se snaží vstoupit do webové aplikace.

Pro zvýšení bezpečnosti a snížení pravděpodobnosti odcizení přihlašovacích údajů bude provedena konfigurace využívající aplikačního protokolu *HTTPS*. Bude zde objasněn proces při vytváření certifikátů jak *self-signed*, tak pro certifikační autoritu, které jsou vyžadovány pro nasazení webových aplikací do Internetu.

V diplomové práci je použit *rpm* Apache určený pro oba typy architektur (32bit a 64bit).

```
yum install httpd.x86_64
```

Po zdárné instalaci jsou v */etc/httpd/* dostupné konfigurační soubory Apache. Hlavní konfigurační soubor je umístěn v adresáři *conf* a nachází se v souboru *httpd.conf*.

```
[root@ansible-server ~]# cd /etc/httpd/
[root@ansible-server httpd]# ll
total 8
drwxr-xr-x. 2 root root   57 Mar  3 07:50 conf
drwxr-xr-x. 2 root root 4096 May  1 21:22 conf.d
drwxr-xr-x. 2 root root 4096 Apr  6 14:26 conf.modules.d
lrwxrwxrwx. 1 root root   19 Feb 25 16:38 logs -> ../../var/log/httpd
lrwxrwxrwx. 1 root root   29 Feb 25 16:38 modules -> ../../usr/lib64/httpd/modules
lrwxrwxrwx. 1 root root   10 Feb 25 16:38 run -> /run/httpd
[root@ansible-server httpd]# ll conf.d/
```

Obr. A.60: Adresář obsahující konfigurační soubory Apache.

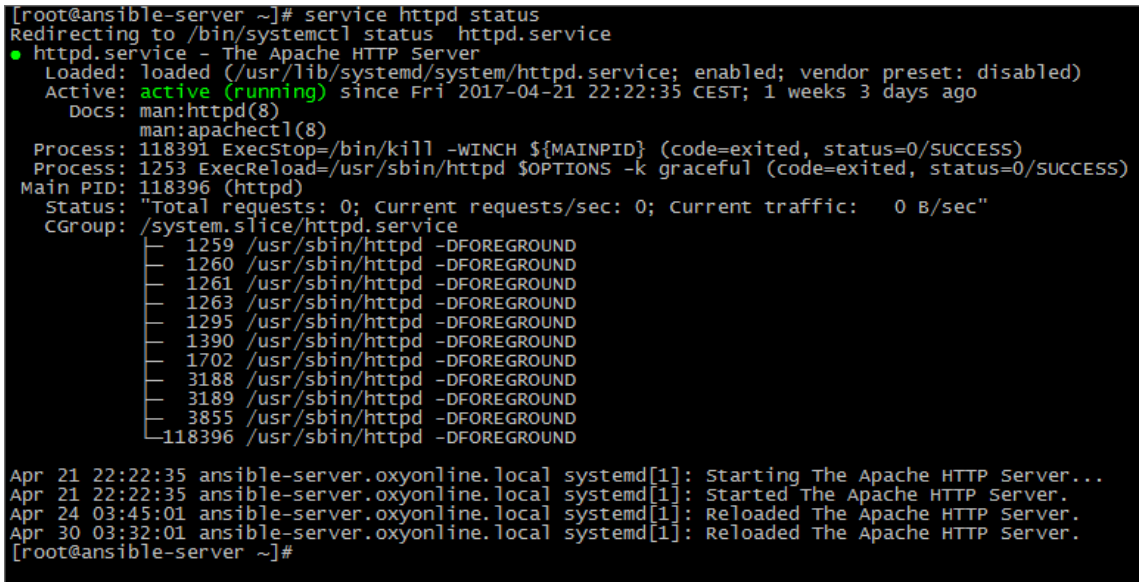
Nyní bude práce rozdělena na konfiguraci webových služeb pomocí HTTP a HTTPS protokolu. Před samotnou konfigurací jednotlivých protokolů je zapotřebí spustit a povolit službu Apache.

```
systemctl start httpd
```

```
systemctl enable httpd
```

Ověření běhu Apache.

```
service httpd status
```



Obr. A.61: Běžící služba Apache.

- **Konfigurace HTTP**
  Apache je ihned po instalaci a provedení základních úkonů (povolení, spuštění služby a nastavení firewallu, Kapitola A.4.2) schopen odpovídat na požadované dotazy. Stále však není nastaven, aby dotazy směroval do webové aplikace.
  Aby byly požadavky směrovány přímo do webové aplikace umožní direktiva *ProxyPass* zapsána na konec konfiguračního souboru *httpd.conf* v tomto znění.

```
ProxyPass / http://localhost:8080/ timeout=900 keepalive=On
```

  Lomítkem za direktivou je řečeno, pokud uživatel použije v URL adrese pouze název domény (http://vut-ansible.cz) bude požadavek přesměrován na adresu 127.0.0.1 (localhost) s portem 8080, kde naslouchá springová aplikace. Aplikace vrátí data určená pro URL adresu *"/"* zaslána metodou *GET*. Parametrem *timeout* je nastaven čas při kterém Apache nesmí ukončit spojení. Pokud je odpověď doručena uživateli ve stanoveném čase, spojení může být ukončeno.

  Dalším důležitým parametrem direktivy *ProxyPass* je *keepalive*. Je využit pokud se nachází mezi webovým serverem a klientem firewall. Pokud se vyskytne v TCP relaci neaktivní spojení, firewall jej ihned ukončí. Tímto vyvstane velký problém. Pokud jsou spuštěny rozsáhlejší playbooky, tak server odpovídá s časovým zpožděním a relace je tak předčasně ukončena. Vznikne tak nežádoucí stav, kdy je odpověď serveru ukončena chybovým hlášením. Z tohoto důvodu je parametr nastaven na *on*, tím je server upozorněn, aby při možném stavu

ukončení spojení začal zasílat zprávy *KEEP_ALIVE*. Firewall tak neukončí spojení a server má dostatek času zpracovat požadavek a odpovědět. Pro obousměrnou komunikaci je přidán řádek s direktivou *ProxyPassReverse*.

```
    ProxyPassReverse / http://localhost:8080/
```

Konfigurace uvnitř *httpd.conf* vypadá následovně.

```
#
# EnableMMAP and EnableSendfile: On systems that support it,
# memory-mapping or the sendfile syscall may be used to deliver
# files.  This usually improves server performance, but must
# be turned off when serving from networked-mounted
# filesystems or if support for these functions is otherwise
# broken on your system.
# Defaults if commented: EnableMMAP On, EnableSendfile Off
#
#EnableMMAP off
EnableSendfile on

# Supplemental configuration
#
# Load config files in the "/etc/httpd/conf.d" directory, if any.
#IncludeOptional conf.d/*.conf

#############
#presmerovani na Tomcat

ProxyPass / http://localhost:8080/ timeout=900 Keepalive=On
ProxyPassReverse / http://localhost:8080/
```

Obr. A.62: Konfigurace ProxyPass.

Aby byla konfigurace aktuální, je nezbytné po každém editování konfiguračních souborů restartovat službu. Před samotným restartem je doporučeno provést kontrolu konfiguračních souborů pomocí parametru *configtest*.

```
    service httpd configtest
```

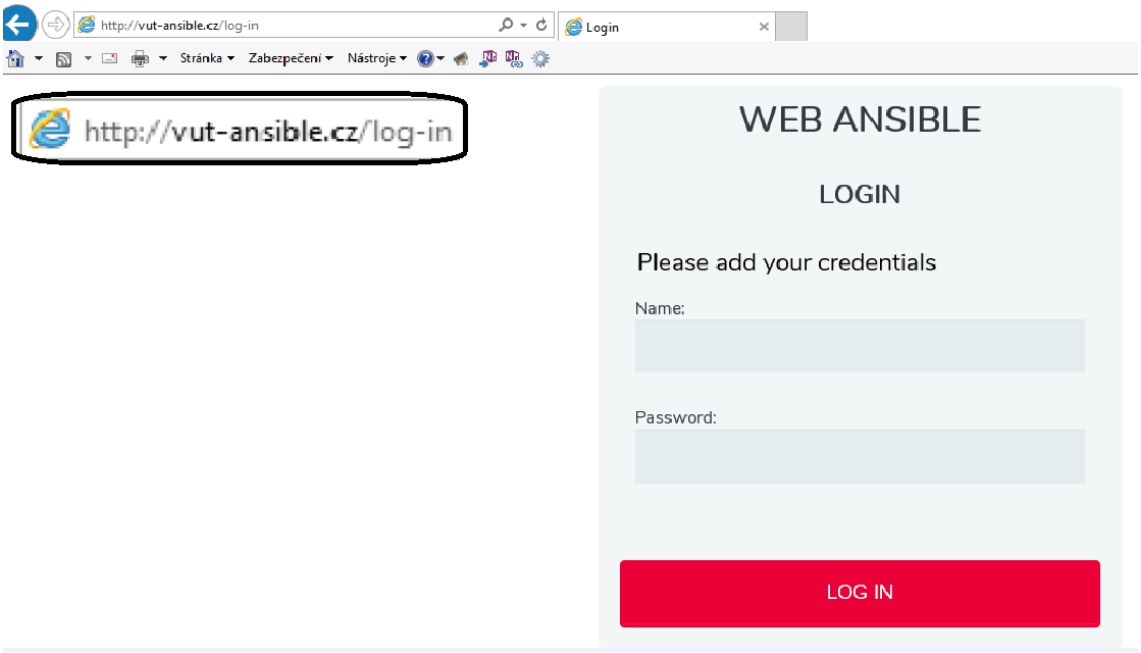
Pokud je zobrazen výpis

```
    Syntax OK
```

může být služba bez problémů restartována.

```
    service httpd restart
```

Pokud je správně nakonfigurován firewall 4.2.2 a ve veřejném DNS 4.2.2 je zaveden záznam o doméně *vut-ansible.cz*, tak by měla být aplikace již dostupná, jak je vidět na Obr. A.63.
Nyní je již aplikace plně funkční a dostupná z veřejné i lokální sítě pod doménou *vut-ansible.cz*.



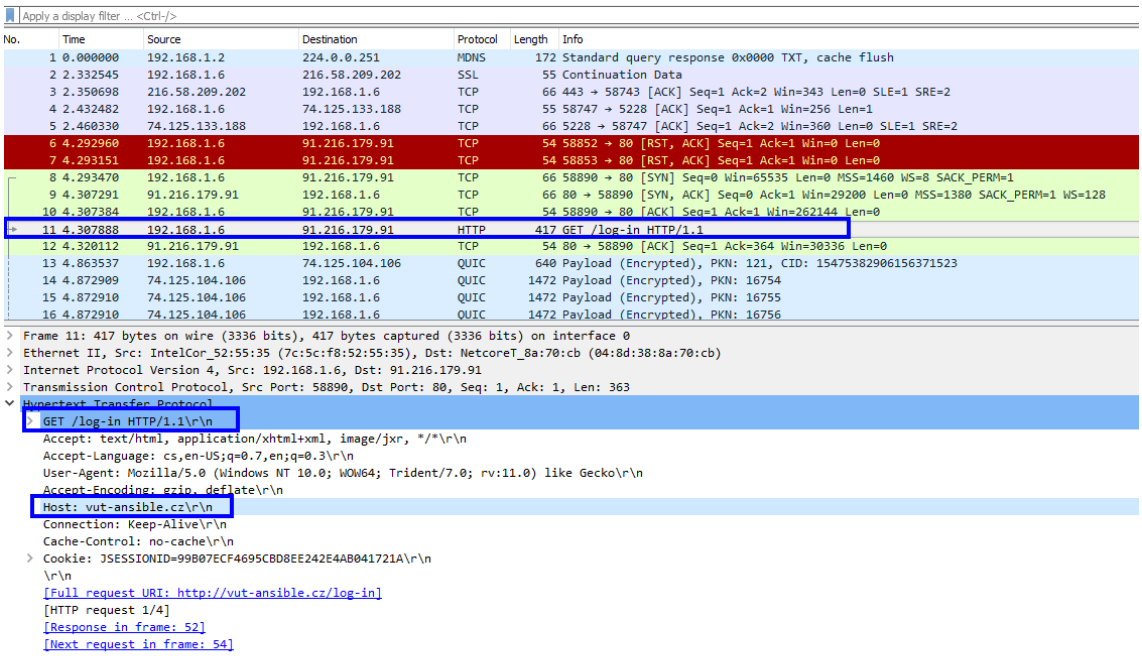
Obr. A.63: Přístup na webovou aplikaci pomocí protokolu HTTP.

Webová aplikace ale není stále dostatečně bezpečná. Pokud by se správce sítě přihlásil do webové aplikace například z počítače který sdílí více uživatelů, mohl by se stát terčem útoku, při kterém by mu byly odcizeny přihlašovací údaje.
Pokud by měl útočník na pozadí spuštěn nějaký síťový analyzátor jako například *Wireshark*, dokázal by z analyzovaných dat zjistit choulostivé informace, jak je demonstrováno na následujících obrázcích Obr. A.64 a Obr. A.65.
Na Obr. A.65 jsou pak jasně patrné odcizené přihlašovací údaje. Aby bylo možné skrýt důležité údaje před možnými útočníky, je potřeba přenášená data šifrovat. Metoda šifrování pomocí protokolu HTTPS bude představena v následující části textu.

- **Konfigurace HTTPS**
  Pro zajištění bezpečnosti webové aplikace, bude využit protokol *HTTPS*, který zprostředkovává šifrovanou komunikaci. Apache ve výchozím nastavení neumožňuje využití HTTPS, je tedy nutné doinstalovat *mod_SSL*, který tuto funkci zpřístupní.

```
yum install mod_ssl
```

Následující příkaz ověří zda je modul nainstalován, Obr. A.66.



Obr. A.64: Odchycení HTTP protokolu s metodou GET pro doménu vut-ansible.cz.

```
yum info mod_ssl
```

V kořenovém adresáři Apache se objeví nově vytvořený konfigurační soubor */conf.d/ssl.conf*, který je určený pro správu komunikace protokolem *HTTPS*. Dále je do *httpd.conf* souboru automaticky přidáná direktiva *Include*.

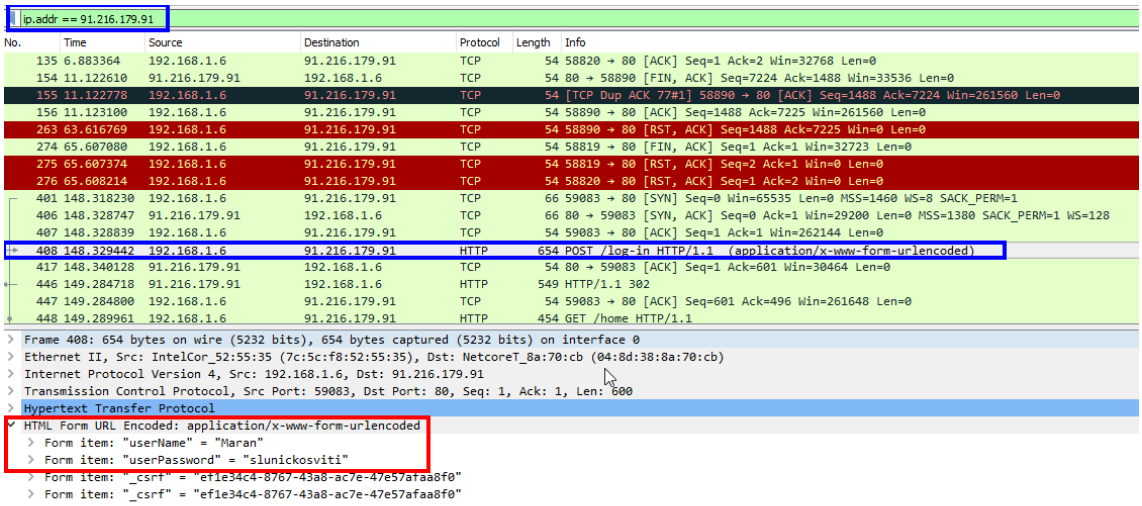
```
Include /etc/httpd/conf.d/ssl.conf
```

Direktiva zpřístupňuje konfiguraci *ssl.conf* přímo uvnitř hlavního konfiguračního souboru. Uvnitř *ssl.conf* jsou přednastaveny všechny direktivy potřebné k zprovoznění šifrované komunikace, jak je zobrazeno na Obr. A.67.
V základu je využit certifikát *localhost.crt*, který je součástí modulu. Není však ověřený certifikační autoritou a v internetových prohlížečích nebude brán jako validní. Certifikát je zpřístupněn direktivou

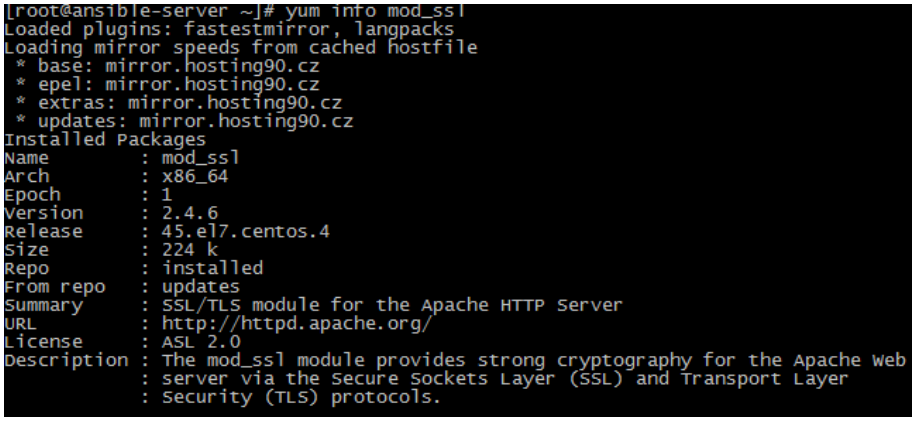
```
SSLCertificateFile /etc/pki/tls/certs/localhost.crt
```

nebyl by ale funkční, pokud by k němu neexistoval privátní klíč *localhost.key*.

```
SSLCertificateKeyFile /etc/pki/tls/private/localhost.key
```



Obr. A.65: Odposlechnutí loginu a hesla pro přístup do webové aplikace.



Obr. A.66: Nainstalovaný modul ssl.

Pokud by již byla na firewallu povolena pravidla pro komunikaci na portu 443, tak by přístup do webové aplikace vypadal následovně, Obr. A.68.
Opět bude analyzována komunikace pomocí nástroje *Wireshark*. Z Obr. A.69 není patrná komunikace pomocí protokolu HTTP jako na Obr. A.65 a co je důležitější, nejsou vidět použité HTTP metody, kterými se požadují (GET) nebo posílají (POST) data. Nemůže tedy dojít k odcizení přihlašovacích údajů, jelikož jsou již data šifrována protokolem *TLSv1.2*. Bližší informace o fungování protokolu *TLS* jsou dostupné na stránce [46].
Komunikace již funguje, ale stále není použit validní certifikát, který by byl ověřen certifikační autoritou. Jak vypadá připojení pomocí validního certifikátu je možné vidět na Obr. 4.8. Ověření certifikátu může být trojího typu:

- **DV (Domain Validation)** – ověří se vlastník domény, na kterou se certifikát vztahuje.
- **OV (Organization Validation)** – ověří se vlastník domény a z nezávislých zdrojů dojde i k ověření existence společnosti, kterou je doména využívána.
- **EV (Extended Validation)** – ověření jako u OV, žadatel navíc získá zelený adresní řádek obsahující název společnosti.

Ve všech případech je nutné, aby žadatel vlastnil doménu. Proces získání domény a práce s DNS záznamy bude popsán v Kapitole 4.2.2. Dále je důležité pro kolik domén je certifikát určen a jakou certifikační autoritou bude ověřován. Stěžejním kriteriem je samozřejmě cena a nabízená garance záruky, proti možnému zneužití certifikátu.
V diplomové práci bude výběr zúžen na ověření domény (DV) certifikační autoritou *SpaceSSL*, kterou je nabízen certifikát *SpaceSSL* umožňující ověření dvou domén pro jediný certifikát, jak je vidět na Obr. A.70. Certifikát tak zajistí doménu s *WWW* předponou i bez ní (**www.vut-ansible.cz**, **vut-ansible.cz**). Tento a mnoho dalších certifikátů je možné získat na stránkach *www.ssls.cz* nebo *www.sslmarket.cz*.

Vhodný certifikát byl vybrán, nyní je potřeba vygenerovat žádost o podepsání certifikátů tzv. soubor *CSR* (Certificate Signing Request). V této žádosti jsou uloženy informace potřebné pro vystavení doménového certifikátu. Jsou jimi:

- **C (Country)** – země ve které se doména nachází,
- **ST (State)** – okres nebo stát,
- **L (Location)** – město,
- **O (Object)** – společnost využívající tuto doménu,
- **OU (Organization Unit)** – organizační jednotka např. IT,
- **CN (Common Name)** – název domény.

Nejdůležitějším parametrem je v tomto případě *Common Name*, určující název domény, pro kterou bude certifikát vystaven. Ostatní položky pro *DV* ověření nejsou klíčové. Příkazem níže bude vygenerován *CSR* soubor s žádostí *vut-ansible.cz.csr* spolu se souborem obsahujícím privátní klíč *vut-ansible.cz.key*.

```
openssl req -new -newkey rsa:2048 -nodes -sha256
```

```
-out vut-ansible.cz.csr -keyout vut-ansible.cz.key
```

```
-subj "/C=CZ/CN=www.vut-ansible.cz"
```

Obsah vygenerované žádosti je potřeba zkopírovat do administračního okna při nákupu, jak demonstruje Obr. A.71. Zároveň je potřeba vybrat metodu ověření, v tomto případě bude vlastnictví domény ověřeno pomocí DNS záznamu (bude předvedeno v Kapitole A.4.2).
Po zdárném ověření domény lze z administrace portálu *ssls.cz* stáhnout soubor ve kterém je obsažen certifikát Obr. A.72.
Výsledný certifikát je složen ze tří souborů.

- **ServerCertificate.crt** – kořenový certifikát (vygenerován CA).
- **Intermediate_CA_chain.crt** – soubor řetězící dodatečné informace ke kořenovému certifikátu (vygenerován CA).
- **vut-ansible.cz.key** – privátní klíč kořenového certifikátu.

Soubory jsou klíčovými prvky pro správnou funkci certifikátu a je potřeba je vložit do úložiště certifikátů Obr. A.73

```
/etc/pki/tls/certs/
```

a úložiště klíčů Obr. A.74.

```
/etc/pki/tls/private
```

Nyní potřeba zaměnit názvy souborů u direktiv *SSLCertificateFile*, *SSLCertificateChainFile* a *SSLCertificateKeyFile* a restartovat službu *httpd*, aby byla do systému načtena aktuální konfigurace.
Po restartu služby bude dostupný ověřený certifikát obsahující informace, které byly zadány při vytváření žádosti *CSR*, jak je patrné na Obr. A.76.

- **Konfigurace HTTP a HTTPS**
  Každý webový prohlížeč zasílá zadané informace protokolem HTTP, pokud není v cestě specifikován protokol HTTPS.

```
https://vut-asnible.cz
```

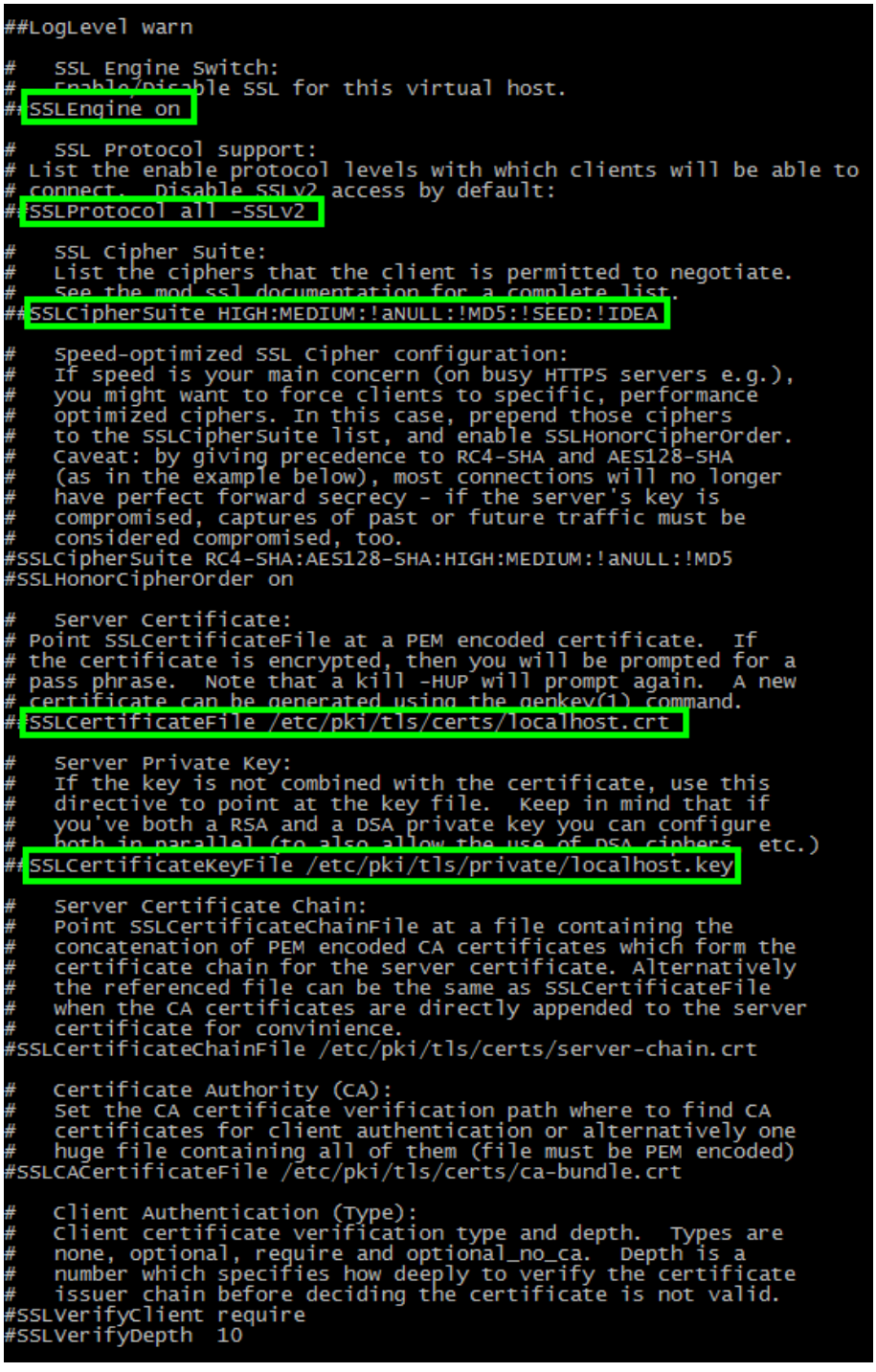
Lze však docílit stavu, kdy po pouhém zadání doménového jména bude komunikace přesměrována z HTTP na HTTPS. Umožňuje to mód *rewrite* s direktivou *RewriteRule*.
Pro docílení tohoto stavu jsou vytvořeny dvě párové direktivy *Virtualhost* a to jak v hlavním konfiguračním souboru *httpd.conf* pro naslouchání na portu

80 (Obr. A.77), tak pro port 443 v souboru *ssl.conf* (Obr. A.78).

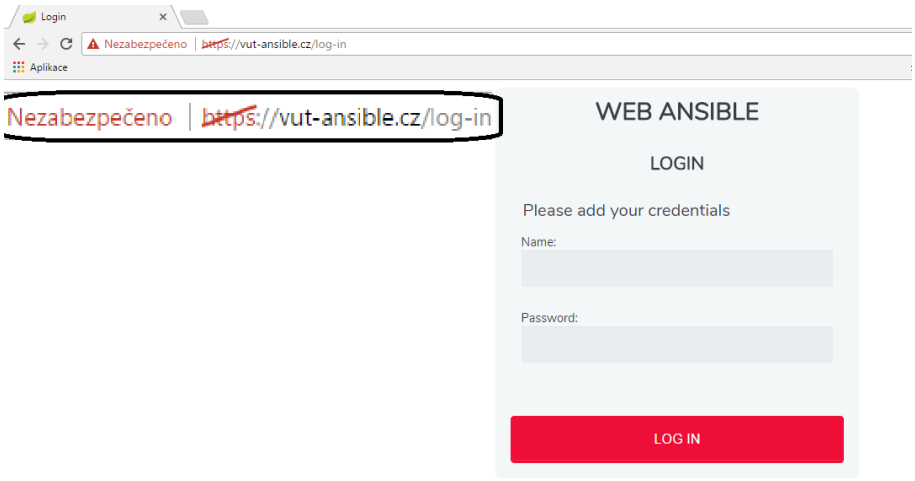
Pokud virtuálnímu hostiteli, naslouchajícímu na portu 80, přijde požadavek *http://vut-ansible.cz*, tak má za úkol tento požadavek přesměrovat na virtuálního hostitele naslouchajícího na portu 443 s doménou *https://vut-ansible.cz*. Takovéto přesměrování řeší direktiva *RewriteRule* s těmito parametry.

```
RewriteRule ^(.*)$      https://vut-ansible.cz$1    [R=301,L]
```

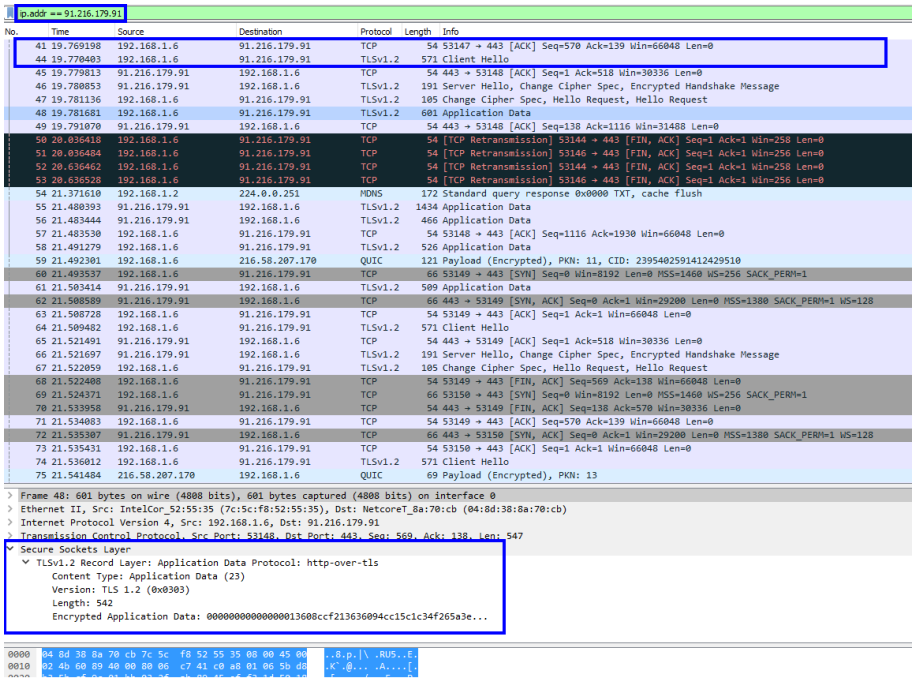
Výsledek znázorňuje Obr. A.79.



Obr. A.67: Část konfiguračního souboru ssl.conf.

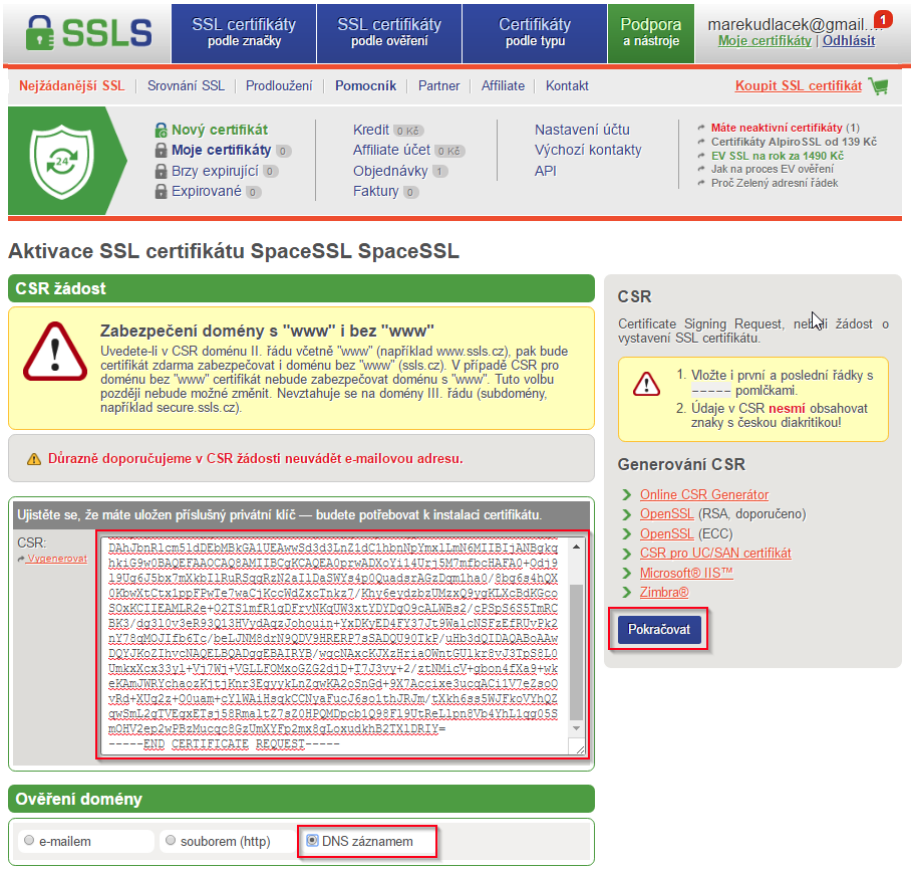


Obr. A.68: Přístup na webovou aplikaci pomocí protokolu HTTPS bez validního certifikátu.



Obr. A.69: Přístup na webovou aplikaci pomocí protokolu HTTPS.

Obr. A.70: DV certifikát od certifikační autority SpaceSSL.



Obr. A.71: Vložení CSR žádosti do nákupní administrace.

Obr. A.72: Stažení .zip souboru s ověřeným certifikátem.

```
[root@ansible-server certs]# cd /etc/pki/tls/certs/
[root@ansible-server certs]# ll
total 36
lrwxrwxrwx. 1 root root   49 Feb 25 16:23 ca-bundle.crt -> /etc/pki/ca-trust/extracted/pem/tls-ca-bundle.pem
lrwxrwxrwx. 1 root root   55 Feb 25 16:23 ca-bundle.trust.crt -> /etc/pki/ca-trust/extracted/openssl/ca-bundle.trust.crt
-rw-r--r--  1 root root 6174 Apr 18 11:24 Intermediate_CA_Chain.crt
-rw-------  1 root root 1517 Apr  6 14:26 localhost.crt
-rwxr-xr-x. 1 root root  610 Feb 20 15:41 make-dummy-cert
-rw-r--r--. 1 root root 2388 Feb 20 15:41 Makefile
-rwxr-xr-x. 1 root root  829 Feb 20 15:41 renew-dummy-cert
-rw-r--r--  1 root root 2045 Apr 18 11:24 ServerCertificate.crt
-rw-r--r--  1 root root 1029 Apr 12 20:45 vut-ansible.cz.csr
```

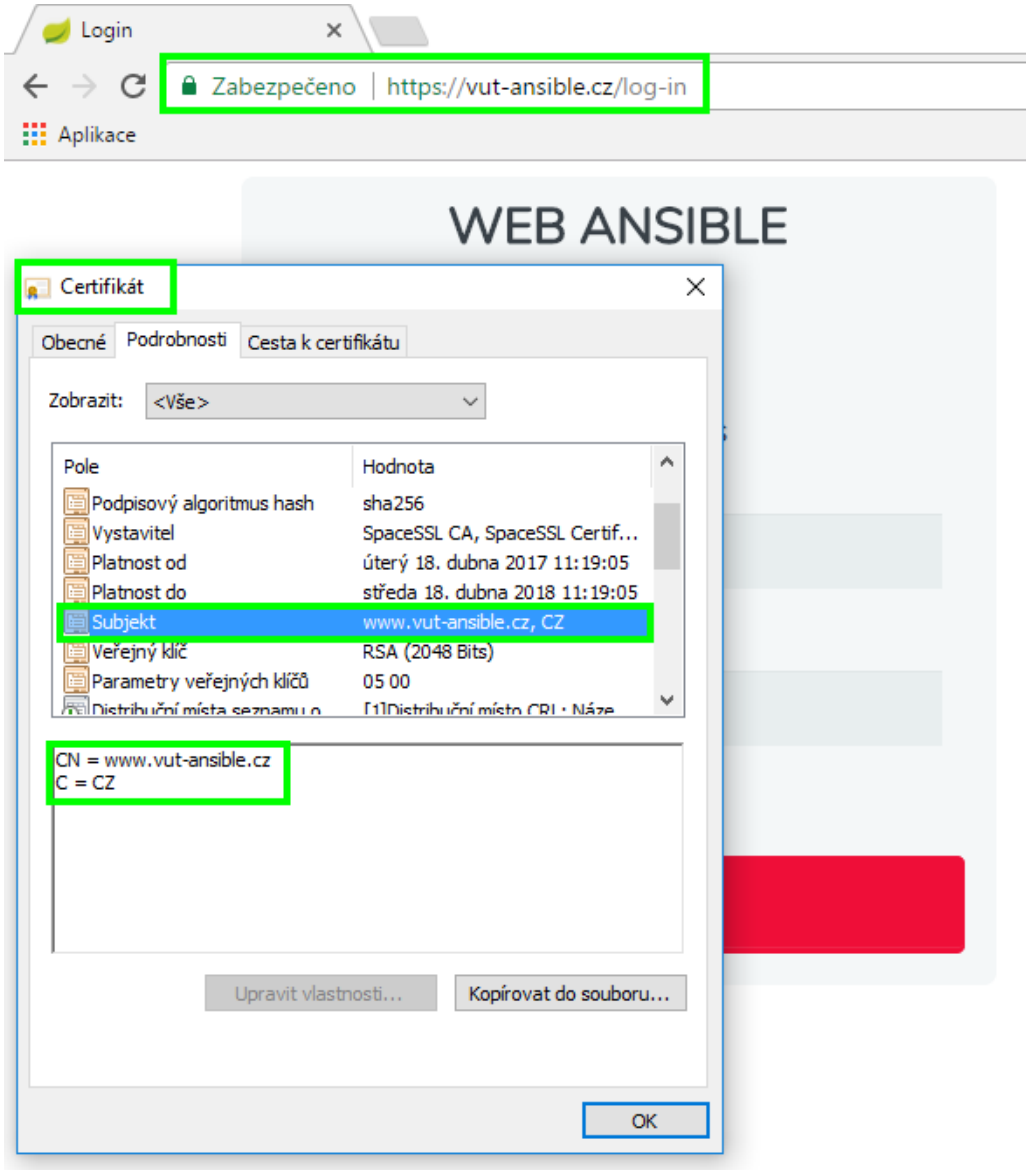
Obr. A.73: Úložiště certifikátů.

```
Last login: Wed May  3 23:36:04 2017 from 192.168.173.232
[root@ansible-server ~]# cd /etc/pki/tls/private/
[root@ansible-server private]# ll
total 8
-rw------- 1 root root 1675 Apr  6 14:26 localhost.key
-rw-r--r-- 1 root root 1704 Apr 18 11:29 vut-ansible.cz.key
[root@ansible-server private]#
```

Obr. A.74: Úložiště klíčů.

```
ServerName vut-ansible.cz
SSLCertificateFile /etc/pki/tls/certs/ServerCertificate.crt
SSLCertificateKeyFile /etc/pki/tls/private/vut-ansible.cz.key
SSLCertificateChainFile /etc/pki/tls/certs/Intermediate_CA_Chain.crt
```

Obr. A.75: Použití ověřeného certifikátu v souboru ssl.conf.



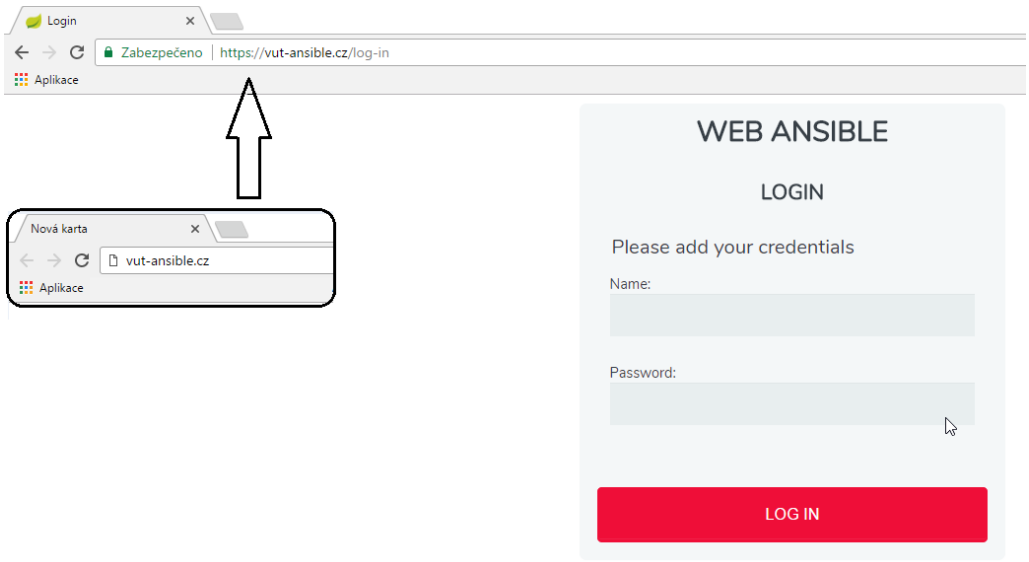
Obr. A.76: Přístup na webovou aplikaci pomocí protokolu HTTPS s validním certifikátem.

```
#############
#presmerovani na Tomcat

ProxyPass / http://localhost:8080/ timeout=900 Keepalive=On
ProxyPassReverse / http://localhost:8080/

<VirtualHost *:80>
ServerName vut-ansible.cz
##DocumentRoot /var/www/html
RewriteEngine On
RewriteRule ^(.*)$      https://vut-ansible.cz$1    [R=301,L]
</VirtualHost>
```

Obr. A.77: Virtualhost naslouchající na portu 80.

```
<VirtualHost *:443>
        ServerAdmin webmaster@vut-ansible.cz
        ServerName vut-ansible.cz
        SSLCertificateFile /etc/pki/tls/certs/ServerCertificate.crt
        SSLCertificateKeyFile /etc/pki/tls/private/vut-ansible.cz.key
        SSLCertificateChainFile /etc/pki/tls/certs/Intermediate_CA_Chain.crt
        SSLEngine on
        SSLHonorCipherOrder on
        SSLProtocol             all -SSLv2 -SSLv3
        SSLCipherSuite          ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-SHA384:
        ErrorLog logs/ssl_error_log
        TransferLog logs/ssl_access_log
        LogLevel warn
        CustomLog logs/ssl_request_log \
          "%t %h %{SSL_PROTOCOL}x %{SSL_CIPHER}x \"%r\" %b"
</VirtualHost>
```

Obr. A.78: Virtualhost naslouchající na portu 443.

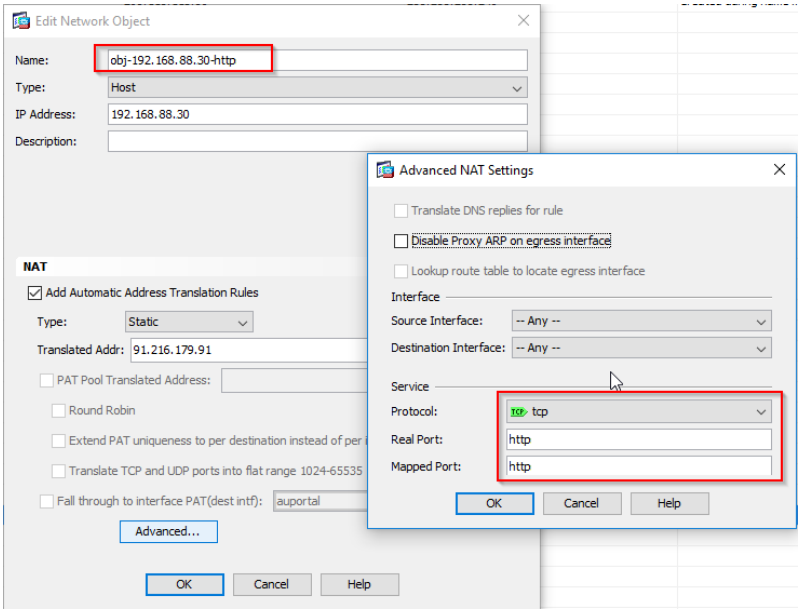


Obr. A.79: Přesměrování komunikace z HTTP na HTTPS.

Firewall a NAT

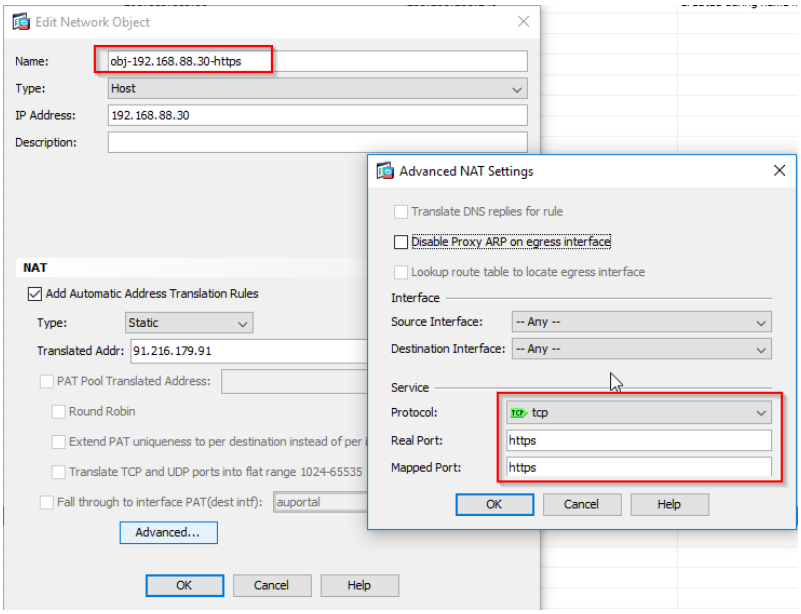
V následujícím textu budou objasněny konfigurace provedené jak na hardwarovém firewallu (*CISCO ASA*), tak na softwarovém firewallu (*firewalld*) obsaženém na Ansible serveru. Účelem konfigurací je správné nastavení pravidel, která umožní průchod paketům určených pro webový server. Jsou to pakety směrované na port 80 (HTTP) a 443 (HTTPS). Současně zde bude prezentováno správné nastavení služby NAT.

- **CISCO ASA 5510**
  Je nutné vytvořit dva síťové objekty *Network Objects* v záložce *Firewall* jak pro port 80, tak pro port 443, využívající transportní protokol *TCP*, jak je znázorněno na obrázcích Obr. A.80 a Obr. A.81.
  V konfiguraci je rovnou povolena funkce NATu, která má za úkol překládat komunikaci z veřejné IP adresy *91.216.179.91*, na lokální IP adresu *192.168.88.30*, která je IP adresou serveru, na kterém je zprovozněn nástroj Ansible spolu s webovou aplikací.



Obr. A.80: Povolení komunikace na portu 80 směrované na Ansible server.

- **firewalld**
  Konfigurací v předešlém kroku je docíleno komunikace s Ansible serverem na portu 80 a 443. I když na serveru běží Apache, který naslouchá na obou portech, tak nebude komunikace stále funkční. Důvodem je systémový firewall *firewalld* (součástí distribuce CentOS7), který má ve výchozím nastavení zakázánu veškerou komunikaci. Příkazy níže vytvoří pravidla, která povolí ko-



Obr. A.81: Povolení komunikace na portu 443 směrované na Ansible server.

munikaci na výše zmíněných portech. Nakonec je potřeba firewall restartovat, aby byla nahrána aktuální konfigurace.

```
firewall-cmd --add-service=http --permanent
```

```
firewall-cmd --add-service=https --permanent
```

Nakonec je potřeba firewall restartovat, aby byla nahrána aktuální konfigurace.

```
service firewalld restart
```

Pro kontrolu si je možné vypsat povolené služby.

```
firewall-cmd --list-services
http ssh https
```

Od této chvíle bude webová aplikace dostupná z veřejné sítě.

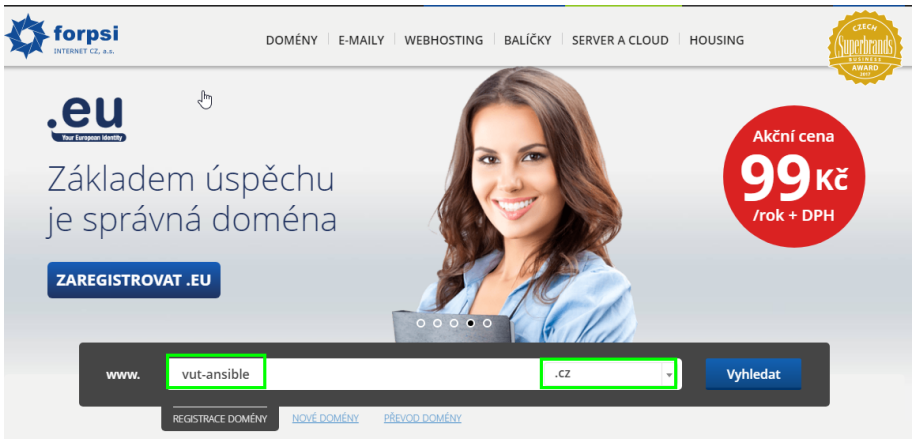
DNS

V textu budou objasněny informace týkající se nákupu domény a její následná správa. Aby bylo možné záznamy editovat je nejprve nutné doménu vlastnit. Na Internetu existuje velké množství poskytovatelů, kteří domény prodávají a zároveň poskytují hosting. V ČR mezi ně patří například portál *active24.cz*, *forpsi.com*, *nic.cz* a spousty dalších. Domény jsou rozlišovány podle úrovní, viz Obr. A.82.

- **První úrovně** – názvy států, oganizace (cz, de, org, com, atd.),
- **Druhé úrovně** – libovolné názvy domén (seznam, facebook, potraviny, atd.)
- **Třetí úrovně** – subdomény specikující nějakou službu (mail, eshop, ftp, atd.)

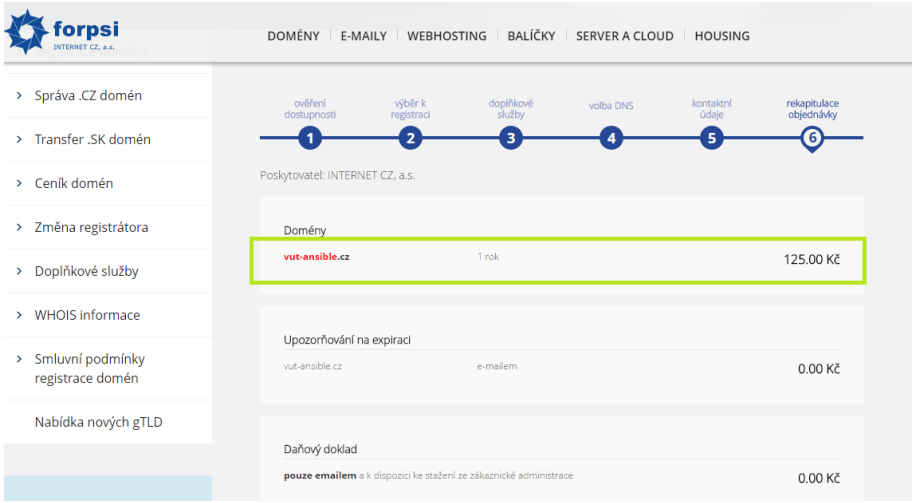
Obr. A.82: Řazení domén podle úrovní.

Vlastníkem domény může být jakákoliv fyzická nebo právnická osoba. Doména *vut-ansible.cz*, která je použita pro přístup na webovou aplikaci, byla pořízena z portálu *forpsi.com*. Níže je vyobrazen postup při nákupu vlastní domény. Nejprve je nutné zjistit zda zvolená doména není již registrována Obr. A.83.



Obr. A.83: Registrace domény vut-ansible.cz.

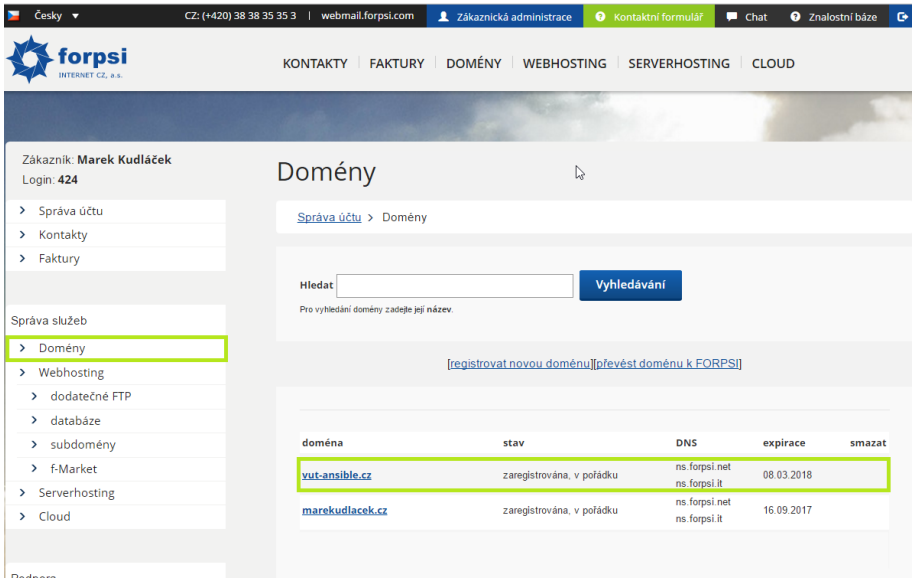
Pokud není doména registrována, lze jí koupit Obr. A.84. Po nákupu domény je již možné v administraci vidět zakoupenou doménu, Obr. A.85.



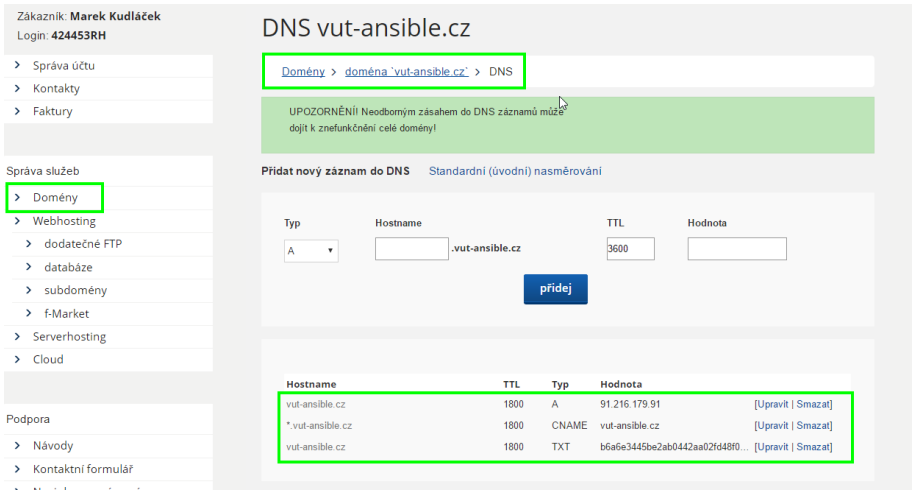
Obr. A.84: Nákup domény vut-ansible.cz.

Aplikace byla spuštěna, na firewallech byla nastavena pravidla, nezbývá než provést poslední krok a přiřadit *A* záznam do systému DNS. Ten umožní nasměrování domény na webový server umístěný v lokální síti. Na Obr. A.87 jsou vidět nastavené DNS záznamy.

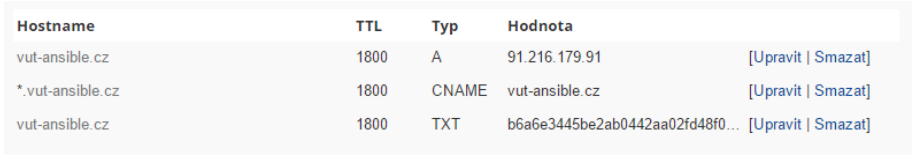
A záznam určuje, na jakou IP adresu má doména směrovat. Záznam *CNAME* umožňuje použití jakékoliv subdomény třetí úrovně s doménou druhé úrovně, kdy výsledek bude vždy přesměrován na doménu *vut-ansible.cz*. Poslední záznam *TXT*, byl využit pro ověření vlastnictví domény. Vlastnictví domény bylo ověřováno certifikační autoritou před vystavením validního certifikátu, jak bylo probráno v Kapitole A.4.2.



Obr. A.85: Administrační rozhraní forpsi.com.



Obr. A.86: Nastaveni DNS záznamů.



Obr. A.87: DNS záznamy.

LITERATURA

[1] CFEngine, "Cfengine," https://cfengine.com/, (Accessed on 10/8/2016).

[2] Puppet, "Puppet - the shortest path to better software," https://puppet.com/, (Accessed on 10/8/2016).

[3] Chef, "Chef – automate your infrastructure | chef," https://www.chef.io/chef/, (Accessed on 10/8/2016).

[4] Ansible, "Ansible is simple it automation," https://www.ansible.com/, (Accessed on 10/8/2016).

[5] Puppet, "Installing puppet: Pre-install tasks — documentation — puppet," https://docs.puppet.com/puppet/latest/reference/install_pre.html, (Accessed on 10/8/2016).

[6] ——, "Installing puppet agent: Microsoft windows — documentation — puppet," https://docs.puppet.com/puppet/4.7/reference/install_windows.html, (Accessed on 10/8/2016).

[7] ——, "Index of /windows," https://downloads.puppetlabs.com/windows/, (Accessed on 10/8/2016).

[8] ——, "Puppet forge," https://forge.puppet.com/, (Accessed on 10/8/2016).

[9] ——, "Resource type reference (single-page) — documentation — puppet," https://docs.puppet.com/puppet/latest/reference/type.html, (Accessed on 10/8/2016).

[10] Ansible, "Installation — ansible documentation," http://docs.ansible.com/ansible/intro_installation.html, (Accessed on 10/8/2016).

[11] ——, "7. requirements — ansible tower installation and reference guide," http://docs.ansible.com/ansible-tower/2.2.2/html/installandreference/requirements_refguide.html, (Accessed on 10/8/2016).

[12] ——, "Module index — ansible documentation," http://docs.ansible.com/ansible/modules_by_category.html, (Accessed on 10/8/2016).

[13] Chef, "Install the chef server — chef docs," https://docs.chef.io/install_server.html, (Accessed on 10/23/2016).

[14] ——, "System requirements — chef docs," https://docs.chef.io/chef_system_requirements.html, (Accessed on 10/23/2016).

[15] ——, “Chef server | chef downloads | chef,” https://downloads.chef.io/chef-server/, (Accessed on 10/23/2016).

[16] ——, “Install the chef-client on microsoft windows — chef docs,” https://docs.chef.io/install_windows.html, (Accessed on 10/23/2016).

[17] ——, “knife windows — chef docs,” https://docs.chef.io/plugin_knife_windows.html, (Accessed on 10/23/2016).

[18] ——, “About resources — chef docs,” https://docs.chef.io/resource.html, (Accessed on 10/23/2016).

[19] Ansible, “Windows support — ansible documentation,” http://docs.ansible.com/ansible/intro_windows.html#windows-system-prep, (Accessed on 11/28/2016).

[20] Microsoft, “Installing the .net framework,” https://msdn.microsoft.com/en-us/library/5a4x27ek(v=vs.110).aspx, (Accessed on 11/28/2016).

[21] ——, “Download windows management framework 5.0 from official microsoft download center,” https://www.microsoft.com/en-us/download/details.aspx?id=50395, (Accessed on 11/28/2016).

[22] Ansible, “Windows modules — ansible documentation,” http://docs.ansible.com/ansible/list_of_windows_modules.html, (Accessed on 10/8/2016).

[23] Chocolatey, “Chocolatey gallery | packages,” https://chocolatey.org/packages, (Accessed on 12/09/2016).

[24] ——, “win_chocolatey - installs packages using chocolatey — ansible documentation,” http://docs.ansible.com/ansible/win_chocolatey_module.html, (Accessed on 12/09/2016).

[25] Ansible, “win_package - installs/uninstalls an installable package, either from local file system or url — ansible documentation,” http://docs.ansible.com/ansible/win_package_module.html, (Accessed on 12/10/2016).

[26] ——, “win_reboot - reboot a windows machine — ansible documentation,” http://docs.ansible.com/ansible/win_reboot_module.html, (Accessed on 12/10/2016).

[27] ——, “shell - execute commands in nodes. — ansible documentation,” http://docs.ansible.com/ansible/shell_module.html, (Accessed on 12/10/2016).

[28] ——, "win_shell - execute shell commands on target hosts. — ansible documentation," http://docs.ansible.com/ansible/win_shell_module.html, (Accessed on 12/10/2016).

[29] ——, "win_user - manages local windows user accounts — ansible documentation," http://docs.ansible.com/ansible/win_user_module.html, (Accessed on 12/10/2016).

[30] ——, "user - manage user accounts — ansible documentation," http://docs.ansible.com/ansible/user_module.html, (Accessed on 12/10/2016).

[31] ——, "win_file - creates, touches or removes files or directories. — ansible documentation," http://docs.ansible.com/ansible/win_file_module.html, (Accessed on 12/11/2016).

[32] ——, "file - sets attributes of files — ansible documentation," http://docs.ansible.com/ansible/file_module.html, (Accessed on 12/11/2016).

[33] ——, "lineinfile - ensure a particular line is in a file, or replace an existing line using a back-referenced regular expression. — ansible documentation," http://docs.ansible.com/ansible/lineinfile_module.html, (Accessed on 12/11/2016).

[34] ——, "win_lineinfile - ensure a particular line is in a file, or replace an existing line using a back-referenced regular expression. — ansible documentation," http://docs.ansible.com/ansible/win_lineinfile_module.html, (Accessed on 12/11/2016).

[35] ——, "win_updates - download and install windows updates — ansible documentation," http://docs.ansible.com/ansible/win_updates_module.html, (Accessed on 12/11/2016).

[36] "Getting started - vagrant by hashicorp," https://www.vagrantup.com/docs/getting-started/, (Accessed on 02/19/2017).

[37] "Vagrant box centos/7 | atlas by hashicorp," https://atlas.hashicorp.com/centos/boxes/7/, (Accessed on 02/19/2017).

[38] "Spring tool suite™ (sts)," https://spring.io/tools/sts, (Accessed on 04/22/2017).

[39] "Spring documentation," https://spring.io/docs, (Accessed on 04/23/2017).

[40] "Java se - downloads | oracle technology network | oracle," http://www.oracle.com/technetwork/java/javase/downloads/index.html, (Accessed on 04/23/2017).

[41] “Maven model – maven,” https://maven.apache.org/ref/3.5.0/maven-model/maven.html, (Accessed on 04/24/2017).

[42] “Stavíme firewall (1) - root.cz,” https://www.root.cz/clanky/stavime-firewall-1/, (Accessed on 05/04/2017).

[43] “Stavíme firewall (2) - root.cz,” https://www.root.cz/clanky/stavime-firewall-2/, (Accessed on 05/04/2017).

[44] “Getting started · serving web content with spring mvc,” https://spring.io/guides/gs/serving-web-content/, (Accessed on 04/24/2017).

[45] “Tools,” https://spring.io/tools, (Accessed on 04/24/2017).

[46] “Transport layer security - wikipedia,” https://en.wikipedia.org/wiki/Transport_Layer_Security, (Accessed on 05/03/2017).